MICROCONTRÔLEURS
ET ENVIRONNEMENT

Circuits logiques programmables

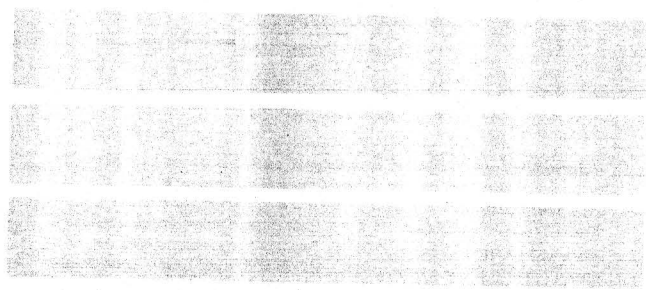
CHRISTIAN TAVERNIER

DUNOD

CIRCUITS LOGIQUES PROGRAMMABLES

CHRISTIAN TAVERNIER

Nouvelle édition



DUNOD

Ce pictogramme mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire, le développement massif du ***photocopillage***.

Le Code de la propriété intellectuelle du 1er juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée dans les établissements d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée.

Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation du Centre français d'exploitation du droit de copie (**CFC**, 3 rue Hautefeuille, 75006 Paris).

© Dunod, Paris, 1996

ISBN 2 10 002882 0

Toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite selon le Code de la propriété intellectuelle (Art L 122-4) et constitue une contrefaçon réprimée par le Code pénal. • Seules sont autorisées (Art L 122-5) les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective, ainsi que les analyses et courtes citations justifiées par le caractère critique, pédagogique ou d'information de l'œuvre à laquelle elles sont incorporées, sous réserve, toutefois, du respect des dispositions des articles L 122-10 à L 122-12 du même Code, relatives à la reproduction par reprographie.

CIRCUITS LOGIQUES PROGRAMMABLES

TABLE DES MATIÈRES

AVANT-PROPOS 5

1 NOTIONS ESSENTIELLES 7
Organisation de l'ouvrage 9
Classification générale 10
Programmateur et système de développement 18

2 LES MÉMOIRES MORTES PROGRAMMABLES OU PROM 23
Les PROM bipolaires à fusibles 25
Les UVPROM ou PROM effaçables aux ultra-violets 33
Les One Time PROM ou OTP 53
Les PROM effaçables électriquement ou EEPROM 54
Les mémoires flash ou flash EPROM 59
Utilisation des mémoires comme circuits logiques 70

3 LES PAL BIPOLAIRES 77
Notions de base 79
Les PAL combinatoires et à registres classiques 94
Les PAL versatiles ou V PAL 105
Un exemple d'application 110

4 LES GAL ET PAL CMOS 115
Notions de base 118
Les GAL ou PAL CMOS classiques 119
Le GAL ou PAL CMOS asynchrone à registre 20RA10 127
L'ispGAL 22V10 130
Les GAL 6001 et 6002 ou GAL à architecture de type FPLA 134

5 LES EPLD 141
Les EPLD Altera 145
Les EPLD MACH d'AMD 181
Les EPLD pLSI et ispLSI de Lattice 190

6 LES FPGA DE TYPE RAM 199
Les LCA de Xilinx 201
Les Flex d'Altera 232

7 LES FPGA À ANTI-FUSIBLES 247
Notions de base 249
Les différentes familles de FPGA à anti-fusibles 260

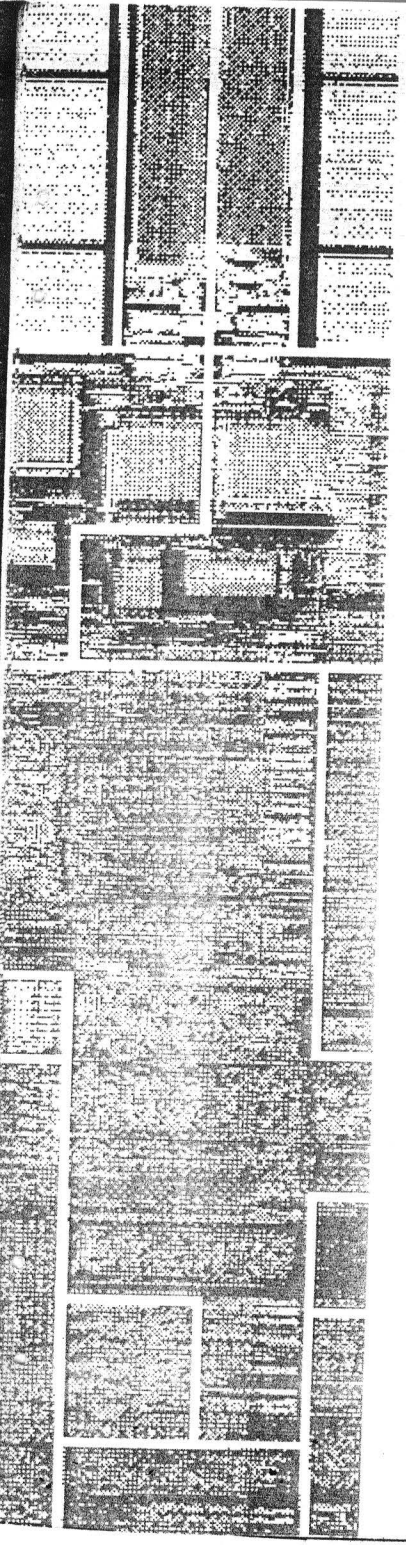
CIRCUITS LOGIQUES PROGRAMMABLES

AVANT PROPOS

Les circuits logiques programmables par l'utilisateur occupent une place prépondérante dans les appareils électroniques modernes. On les trouve sous forme de mémoires bien sûr, dans les micro-ordinateurs, mais aussi et surtout sous forme de circuits logiques à part entière pour remplacer des sous-ensembles qui, par le passé, auraient utilisé des dizaines voire des centaines de boîtiers logiques conventionnels.

L'offre des fabricants en ce domaine est vaste et diversifiée, et s'étend des boîtiers ne contenant que quelques dizaines de portes à des circuits intégrant sans problème plus de 20000 portes logiques. Malgré cela, ils restent programmables sur des systèmes de développement simples d'emploi et à la portée de tous ; systèmes qui ne requièrent très souvent qu'une formation très brève de l'utilisateur tant ils sont bien conçus et efficaces.

En outre, ces circuits sont proposés en diverses technologies couvrant un maximum de besoins : circuits programmables une seule fois, circuits effaçables aux ultraviolets, circuits programmables et effaçables électriquement et même circuits dont la configuration peut être modifiée en cours d'utilisation.



ORGANISATION DE L'OUVRAGE

Les mémoires et les circuits logiques programmables sont susceptibles d'intéresser de nombreux utilisateurs de tous horizons et donc, nécessairement, de niveaux de connaissances de départ très divers. Aussi, afin que notre ouvrage puisse être lu par tous avec un maximum de profit, avons-nous adopté un découpage aussi pédagogique et progressif que possible. Ainsi, ceux d'entre vous qui connaissent déjà certains circuits présentés dans ce livre peuvent éventuellement sauter les parties qu'ils connaissent. Voici donc en quelques lignes comment tout cela est organisé.

Ce chapitre devrait à notre avis être lu par tous. Il propose en effet une classification générale de ces circuits et clarifie quelque peu la signification des innombrables sigles que l'on rencontre dans le monde des circuits logiques programmables. Sa lecture nous permet ensuite de parler un seul et même langage. Nous avons en effet privilégié au maximum l'utilisation des appellations les plus utilisées ou les plus connues, même si elles n'étaient pas toujours les plus judicieuses.

Le deuxième chapitre est consacré exclusivement aux PROM ou mémoires programmables par l'utilisateur. Il forme un tout qui peut être lu si ces produits vous intéressent ou être ignoré si vous ne vous intéressez qu'aux circuits logiques proprement dits.

Le troisième chapitre est consacré aux PAL. Même si ces circuits sont actuellement en voie de disparition au profit de produits plus performants en termes de densité d'intégration et de souplesse de programmation, ce sont les ancêtres de tous les circuits logiques programmables actuels. Les notions qui y sont présentées doivent être lues par tous car il y est forcément fait référence dans la suite de l'ouvrage.

Les chapitres suivants, consacrés chacun à une famille logique différente, peuvent être abordés dans l'ordre de votre choix. L'ordre le plus logique est celui du livre, bien sûr, mais si vous avez un besoin pressant de savoir ce qu'est un LCA, vous pouvez aller directement du chapitre 3 au chapitre 6 sans difficulté.

De nombreuses appellations diverses ayant vu le jour depuis qu'existent les circuits logiques programmables, vous trouverez, sur le tableau de classification générale de la figure 1.2, ces principales appellations sous la rubrique les concernant. Ceci vous permettra donc, si nécessaire, de vous reporter immédiatement au paragraphe de l'ouvrage qui les étudie.

CLASSIFICATION GÉNÉRALE

Avant de voir comment fonctionnent les différents circuits logiques programmables présents sur le marché et ce qu'on peut en attendre, il est indispensable de réaliser une classification rigoureuse. En effet, une certaine méconnaissance de ces produits alliée à l'arrivée massive de nouveaux termes, tous dérivés de l'américain vu l'origine des circuits, ont conduit à créer une certaine confusion dans les esprits. Pourtant, si l'on prend la peine d'expliquer ce que signifient les différents sigles, la situation devient très vite beaucoup plus claire.

Plutôt que de vous présenter tout de go un organigramme en vous demandant de l'admettre comme tel, nous allons le construire ensemble afin que vous puissiez constater qu'il répond à une certaine logique technique et/ou historique.

Tout d'abord il faut distinguer deux grandes familles de circuits logiques programmables :

- les mémoires mortes programmables désignées sous le sigle générique de PROM ce qui signifie *Programmable Read Only Memory* ou mémoire programmables à lecture seule ;
- les circuits logiques programmables en tant qu'entités circuits logiques qui sont désignés par le sigle générique de PLD ce qui signifie *Programmable Logic Device* pour... circuits logiques programmables, bien sûr.

Les microcontrôleurs, que l'on peut assimiler à des circuits logiques programmables, sont en fait à classer avec les PROM car leur possibilité de programmation tient uniquement à la présence sur la puce de ces circuits d'une PROM. Ils ne sont pas traités dans le cadre de cet ouvrage car ils nécessitent à eux seuls un très long développement et un autre titre de cette même collection leur est entièrement consacré (Les microcontrôleurs 4 et 8 bits du même auteur.)

Les PROM peuvent être utilisées en tant que circuits logiques comme nous le verrons au chapitre 2, même si ces utilisations sont de moins en moins fréquentes, vu la banalisation des *vrais* circuits logiques programmables. Les PLD, par contre, ne peuvent jamais être utilisés en tant que PROM.

LES DIFFÉRENTES FAMILLES DE PROM

Au sein des PROM, il faut distinguer à nouveau deux familles qui ne sont pas faciles à trier, en se basant uniquement sur les appellations, car celles-ci sont assez mal choisies comme vous allez le constater. On rencontre en effet :

- les ROM ou ROM programmables par masque qui ne peuvent être programmées que lors de leur fabrication par action directe sur le masque de gravure du circuit intégré. Ces mémoires sont donc réservées à des productions de grande série et ne peuvent en aucun cas être programmées directement, même avec un appareillage sophistiqué, par leur utilisateur. Ces mémoires ne nous intéressent donc pas dans le cadre de cet ouvrage.
- les PROM qui s'appelaient initialement FPROM, cette appellation ayant disparu à tort car elle signifiait *Field Programmable ROM* c'est-à-dire ROM pouvant être programmée sur site. Ces FPROM ou PROM sont donc des mémoires mortes qui peuvent être programmées par l'utilisateur au moyen d'un appareil adéquat. Il est donc possible de les utiliser pour des productions en petite série voire même à l'unité. Le programmateur, très coûteux s'il doit pouvoir couvrir de nombreuses références de non moins nombreux fabricants, peut être d'un prix beaucoup plus abordable s'il est destiné à une famille particulière de PROM.

Dans la famille des PROM on distingue à nouveau trois familles principales :

- les PROM ou PROM à fusibles (ici aussi le terme PROM est souvent utilisé seul pour les PROM à fusibles ce qui prête à confusion) qui sont des mémoires dont la programmation se réalise par destruction physique d'un ou plusieurs fusibles contenus sur la puce de silicium. Du fait d'une telle action physique, un retour en arrière n'est pas possible et ces mémoires sont donc programmées une fois pour toutes ;
- les EPROM ce qui signifie Erasable PROM c'est-à-dire mémoire programmable et effaçable. Dans de telles mémoire, le processus de programmation est réversible, de diverses manières selon les familles technologiques, et la mémoire programmée peut donc à nouveau être rendue vierge à tout instant si nécessaire ;

- les OTPROM ce qui signifie One Time PROM c'est-à-dire mémoire programmable une fois mais qui ne doivent pas être confondues avec les mémoires à fusibles. En effet, ces OTPROM sont en fait des EPROM dans lesquelles on a inhibé la possibilité d'effacement pour une raison technologique liée généralement au coût de fabrication. Certains auteurs classent d'ailleurs les OTPROM dans les EPROM ce qui procède d'une logique discutable puisque ces mémoires ne sont pas effaçables (même si sur la puce elles en avaient la possibilité).

Dans ces trois familles, seule la famille des EPROM se subdivise à nouveau en trois familles qui sont :

- les UVPROM ou PROM effaçables aux ultraviolets. De telles mémoires se programment électriquement et s'effacent par exposition à un rayonnement ultraviolet de longueur d'onde adéquate. Pour ce faire, leur boîtier est muni d'une fenêtre en quartz caractéristique qui laisse voir la puce et permet son exposition à la lumière ;
- les EEPROM ou E2PROM ce qui signifie *Electricaly Erasable PROM* pour PROM effaçable électriquement. Ces mémoires se programment électriquement comme les UVPROM mais s'effacent, par contre, par simple application de niveaux électriques de tensions et chronogrammes appropriés ;
- les EEPROM FLASH ou FLASH EPROM ou encore mémoires FLASH qui sont des mémoires effaçables et programmables électriquement et qui, à ce titre, pourraient être classées avec les EEPROM normales. Comme la technologie et surtout le mode d'utilisation de ces mémoires est différent, elles méritent une place à part ;
- les NOVRAM, ce qui signifie *NOn Volatile RAM* et qui sont en fait des mémoires vives, d'où le sigle RAM dans l'appellation, rendues artificiellement non volatiles par regroupement dans le même boîtier avec une EEPROM qui recopie en permanence le contenu de la RAM. Ces mémoires peuvent donc être classées avec les RAM ou avec les ROM. Nous les citons ici afin de réaliser une classification complète, mais elles ne seront pas traitées dans le cadre de cet ouvrage car ce ne sont pas de véritables circuits programmables. En outre, leur utilisation est de plus en plus marginale vu la baisse de prix des technologies susceptibles de les concurrencer que sont les EEPROM et les mémoires FLASH.

Attention ! Les vraies NOVRAM que nous venons de décrire sommairement ne doivent pas être confondues avec les RAM non volatiles commercialisées par Dallas, SGS-Thomson ou Benchmarq, par exemple, qui sont en fait de simples RAM classiques dans le boîtier desquelles on a intégré une logique de commande et une pile au lithium de façon à ce qu'elles soient toujours alimentées. Ces *fausses* NOVRAM sont en fait de simples RAM au plan technologique bien sûr.

Dans chaque famille peuvent exister diverses branches qui correspondent à des variantes technologiques ou de programmation, mais la classification vue précédemment est déjà suffisamment précise. Pour bien fixer les idées elle est rappelée figure 1.1.

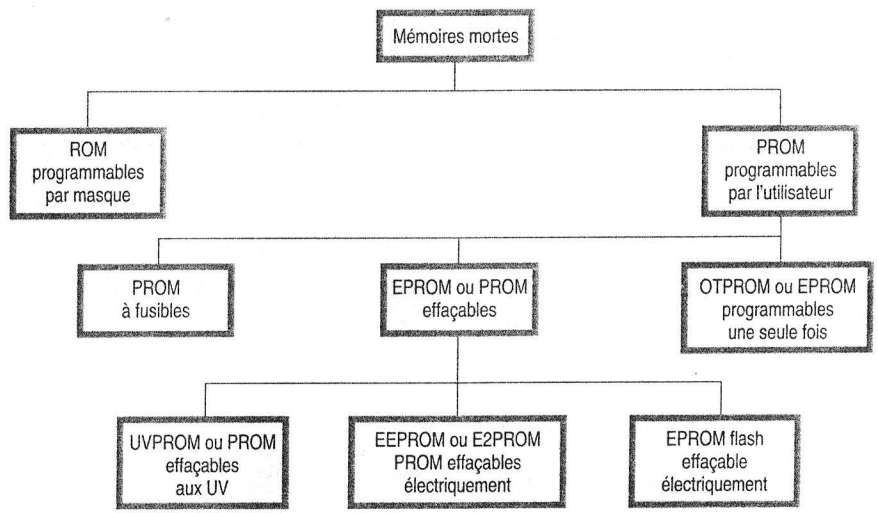


Figure 1.1 :
Classification des principales familles de PROM.

LES DIFFÉRENTES FAMILLES DE PLD

La classification des circuits logiques programmables est un peu plus délicate que celle des ROM ; en effet, les fabricants spécialisés se livrent une bataille féroce pour emporter des parts de marché, et les appellations choisies par certains d'entre eux ne sont pas toujours d'une logique rigoureuse. Nous allons néanmoins essayer de voir un peu plus clair dans la jungle d'appellations que les progrès des technologies d'intégration ne cessent de faire croître.

Les circuits logiques programmables les plus anciens et certainement les plus connus s'appellent les PAL, ce qui est un peu anormal étant donné qu'il s'agit là d'une marque déposée de leur inventeur, la défunte société MMI (qui a maintenant été absorbée par AMD). Comme les réfrigérateurs de la marque Frigidaire, ces PAL ont eu tellement de succès que le terme est passé dans le langage courant malgré son statut de marque déposée. Mot à mot, PAL signifie *Programmable Array Logic*, c'est-à-dire réseau logique programmable. Ce nom semblait donc bien choisi, tout du moins lors de la mise sur le marché de ces circuits, car depuis, il est devenu quelque peu imprécis vu la grande diversité de l'offre.

En fait, il convient aujourd'hui de distinguer les PAL bipolaires, appelés généralement PAL (tout court) et les PAL CMOS qui s'apparentent en fait aux GAL dont nous allons parler ensuite.

Les PAL bipolaires sont toujours programmables par fusibles, comme les PROM du même nom dont ils sont directement issus. Si l'on veut être rigoureux, il faut y distinguer deux sous-familles :

- les PAL combinatoires ou PAL simples qui ne contiennent que des portes d'où l'appellation combinatoire ;
- les PAL à registres qui s'appelaient initialement FPLS ce qui signifiait *Field Programmable Logic Sequencer* pour séquenceur logique programmable sur site. De tels PAL contiennent des registres et peuvent donc faire intervenir dans leur fonctionnement la notion de temps.

Comme les PROM à fusibles, les PAL bipolaires ne sont programmables qu'une fois ce qui peut être gênant en phase de test ou de mise au point d'un appareil, ou bien encore pour le faire évoluer.

Longtemps après la commercialisation du premier PAL à fusibles sont donc arrivés sur le marché des PAL effaçables appelés exclusivement GAL dans un premier temps.

Ces GAL, pour *Generic Array Logic* ou encore réseau logique générique, ce qui ne veut pas dire grand chose, sont aux PAL ce que sont les EEPROM aux PROM. Ils sont en effet programmables et effaçables électriquement. En fait, l'appellation GAL est une marque déposée de la société Lattice Semiconductor qui a été la première à proposer sur le marché ce type de produit. Depuis, les spécialistes traditionnels du PAL que sont par exemple, AMD, Texas, Cypress, se sont réveillés et ont commercialisé eux aussi des circuits analogues aux GAL.

Comme ce terme est déposé, ils ont appelé leurs produits PAL CMOS ou E2PAL ou bien encore PAL EECMOS puisqu'il s'agit en fait de circuits en technologie CMOS programmables et effaçables électriquement. Après une petite guerre commerciale tout à fait normale, les produits et les références se sont homogénéisés pour le plus grand bien de tous et on trouve aujourd'hui des PAL CMOS et des GAL portant les mêmes références génériques et compatibles au plan fonctionnel, même si la technologie de fabrication des circuits n'est pas tout à fait la même.

Parallèlement à cela, Lattice a continué à développer sa famille propre de GAL et propose des références n'ayant, à ce jour, pas d'équivalents chez ses concurrents. Cette société propose en outre des circuits appelés pLSI et surtout ispLSI pour *in system programmable LSI*, c'est-à-dire des circuits que l'on peut programmer (et donc effacer), même lorsqu'ils sont en place sur l'application. Nous leurs consacrons bien sûr une large place dans le chapitre dédié aux EPLD puisque ces circuits sont en fait à mi-chemin entre les EPLD et les FPGA.

À peu près à la même époque que celle de la première commercialisation des GAL, la technologie des PAL CMOS a aussi évolué dans une autre direction sous l'impulsion de la société Altera qui a proposé les premiers PAL CMOS effaçables aux ultraviolets, appelés à l'époque EPLD pour *Erasable Programmable Logic Devices*.

Quel était l'intérêt de tels circuits, effaçables aux UV, alors que les GAL et PAL CMOS étaient effaçables électriquement, nous direz-

vous ? Tout simplement la capacité en nombre de portes et en possibilités de configuration. En effet, alors que les PAL CMOS et les GAL de l'époque étaient limités à quelques dizaines de portes, les premiers EPLD d'Altera atteignaient déjà plusieurs centaines de portes. Ces circuits n'ont pas cessé de s'améliorer depuis comme nous le verrons dans le chapitre de cet ouvrage qui leur est consacré. En particulier, ils existent maintenant en versions effaçables aux ultra-violets, bien sûr, mais aussi en versions effaçables électriquement, que ce soit chez leur promoteur initial Altera, chez Lattice avec ses pLSI et isPLSI ou bien encore chez AMD avec ses familles MACH.

Parallèlement à la famille des PAL, mais bien après la naissance du premier d'entre eux, ont commencé à être commercialisés par les fabricants de circuits intégrés les ASIC sous leurs multiples formes. Ces circuits, qui peuvent être considérés comme les pendants des ROM programmables par masque, ne sont en effet programmables que lors de leur fabrication. Ils sont donc réservés à des productions de très grande série ne serait-ce qu'en raison de leur coût de développement élevé. Ils ne nous intéressent donc pas non plus dans le cadre de cet ouvrage.

Suite à cela, mais il y a quelques années seulement, sont alors apparus les FPGA pour *Field Programmable Gate Array* ou encore réseau de cellules logiques programmables par l'utilisateur. Ces FPGA sont en fait assimilables à des ASIC, mais programmables par l'utilisateur. En effet, ce sont de gros ensembles de blocs logiques élémentaires (plus de 2000 portes) que l'utilisateur peut interconnecter tout à loisir pour réaliser le ou les fonctions logiques de son choix. Deux technologies principales existent à l'heure actuelle pour réaliser ces interconnexions et trois structures de fonctions logiques de base sont en présence.

On distingue en premier les LCA, mis au point initialement par la firme Xilinx, qui utilisent de la mémoire vive statique ou RAM pour configurer les connexions. La famille FLEX d'Altera fait de même pour les interconnexions mais utilise une architecture de circuit et de cellules logiques de base différente.

Enfin, derniers arrivés sur le marché sont les FPGA à antifusibles (mais non, ce n'est pas une mauvaise plaisanterie) proposés initialement par Texas Instruments. Ces FPGA utilisent une technique de programmation des connexions totalement différente, faisant justement appel à des antifusibles. Ces circuits sont programmables électriquement par l'utilisateur, mais ne sont ensuite pas effaçables.

Ceci étant précisé, nous pouvons dresser l'organigramme de la figure 1.2 qui résume la classification que nous venons de réaliser. On constate à l'évidence la prééminence de la technologie CMOS, seule à même d'offrir des densités d'intégration très importantes et surtout la notions de reprogrammabilité, quasiment impossible à obtenir en bipolaire.

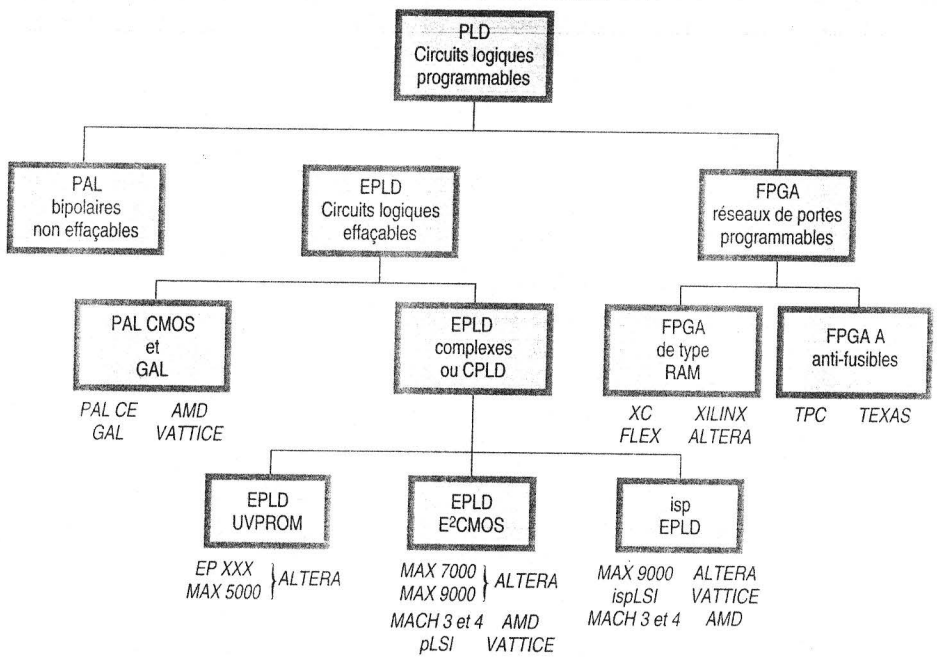


Figure 1.2 :
Classification des principales familles de circuits logiques programmables

PROGRAMMATEUR ET SYSTÈME DE DÉVELOPPEMENT

Qui dit circuit programmable, qu'il soit logique ou non d'ailleurs, dit au minimum programmateur pour les réalisations les plus simples et système de développement pour les réalisations les plus complexes.

Ces sujets sont abordés dans chacun des chapitres suivants car on ne traite pas de la même façon une PROM à fusibles et un LCA. néanmoins, il nous semble opportun de donner ici quelques informations générales communes à tous ces ensembles.

LES PROGRAMMATEURS

Théoriquement, un programmateur de circuit logique peut être dissocié de tout système de développement puisque, en utilisant des données d'entrées fournies dans un format approprié, il n'a pour but que de générer les chronogrammes nécessaires à la programmation du circuit en fonction des spécifications de son fabricant.

En effet, un standard international, le standard JEDEC, a été défini et permet théoriquement à un programmateur qui l'accepte, de comprendre comment il doit programmer tel ou tel circuit en fonction des données qui lui sont fournies sous ce format.

Si cette approche est presque toujours respectée pour les PROM de tous types et pour la majorité des PAL bipolaires simples qui sont des produits relativement standardisés, ce n'est pas toujours le cas pour les EPLD, les GAL et les FPGA. Ce n'est pas forcément dramatique, car pour utiliser efficacement ces trois dernières familles, il vaut mieux faire appel à un système de développement qui comprend quasiment toujours son propre programmateur.

Ceci étant vu, un programmateur peut revêtir plusieurs aspects différents que nous allons définir maintenant.

Les programmateurs autonomes de production

On rencontre en premier lieu les programmateurs autonomes dits de production. Ces appareils sont faits pour programmer le plus vite possible un grand nombre de circuits identiques car, comme leur nom l'indique, ils servent en phase de production d'un matériel. Il faut donc produire beaucoup et vite, mais toujours la même pièce.

Ces programmateurs de production comportent donc généralement plusieurs supports identiques dans lesquels on place les circuits vierges. Ils les programment alors simultanément par recopie d'un circuit maître ou de données qu'on leur a chargé en mémoire au préalable.

Sauf cas particulier, ces appareils ne sont pas prévus pour programmer d'innombrables familles logiques diverses. Ils sont souvent limités à une famille de produits (UVPROM, PAL, etc.) et parfois même à un seul type de produit.

Les programmateurs autonomes de laboratoire

On trouve ensuite les programmateurs autonomes de laboratoire. Ces appareils, en général beaucoup plus coûteux que les précédents, ont pour vocation essentielle le laboratoire d'étude. Ils ne peuvent donc programmer qu'une seule pièce à la fois, mais par contre, acceptent les circuits logiques les plus divers. Le but recherché est de disposer d'un appareil le plus universel possible et surtout, le plus évolutif possible afin qu'il puisse s'adapter sans trop de frais aux nouveaux circuits qui seront commercialisés. Du fait de cette universalité obtenue grâce à des solutions électroniques complexes, ces appareils sont en général assez coûteux. Signalons toutefois l'existence de programmateurs autonomes de laboratoire, beaucoup moins coûteux car destinés à une famille de circuits logiques et une seule. On trouve ainsi des programmateurs pour UVPROM qui supporteront toutes les mémoires de cette famille, mais rien que celles-là ou bien encore des programmateurs de PAL bipolaires. Cette approche est souvent adoptée par de petits laboratoires qui travaillent dans un domaine spécialisé et ne peuvent investir dans un gros programmateur polyvalent.

Les programmateurs non autonomes ou cartes de programmation

Les programmateurs de production et plus encore les programmateurs de laboratoire comportent une partie micro-informatique non négligeable, d'où l'idée d'utiliser pour cela ce qui existe déjà dans un micro-ordinateur en lui ajoutant seulement la circuiterie nécessaire à la production des signaux de programmation proprement dit.

Cette approche a conduit les fabricants à mettre sur le marché des cartes de programmation, essentiellement destinées au monde des compatibles PC qui se prêtent fort bien à cela. De tels programmateurs sont alors constitués par une carte à placer dans un des connecteurs d'extension du PC, reliée à un boîtier externe sur lequel sont placés le ou les supports pour les circuits à programmer.

Un logiciel chargé au moment opportun, pilote cet ensemble et transforme alors le PC en un programmateur souvent très performant. En effet, après quelques cartes simples pour ne pas dire simplistes, commercialisées au début de cette approche, les cartes actuellement mises sur le marché supportent un nombre grandissant de circuits différents et évoluent bien souvent par simple changement de la disquette de logiciel ce qui est souple et peu coûteux.

Bien qu'il soit possible de constituer aussi de la sorte des programmateurs de production, c'est une solution assez peu souvent retenue, car l'encombrement de l'ensemble est relativement important et sa manipulation plus complexe que celle d'un appareil de production autonome. L'approche carte pour PC est donc généralement réservée à la réalisation de programmateurs de laboratoire.

Cette approche est aussi souvent la seule utilisée pour la majorité des circuits logiques les plus complexes, tels les FPGA ou les EPLD par exemple, dont le système de développement sur PC inclut tout ce qu'il faut pour assurer avec un interfaçage minime la programmation directe du circuit comme nous le verrons dans les chapitres consacrés à ces familles de produits.

LES SYSTÈMES DE DÉVELOPPEMENT

Il est évident que la programmation des PROM, qu'elles soient utilisées en tant que telles ou en tant que circuits logiques, ne nécessite aucun système de développement particulier. Bien sûr, si ces dernières contiennent un programme destiné à un microprocesseur, il faut un système de développement pour l'écrire ; cependant, ce système n'est pas lié à la mémoire mais au microprocesseur qui l'utilisera.

Pour ce qui est des PLD, tous types confondus, le recours à un système de développement est presque toujours obligatoire. En effet,

lorsque l'on veut réaliser un circuit logique, on commence par écrire les équations ou tables régissant les divers signaux à traiter mais, alors qu'en logique classique on traduit directement celles-ci en composants réels qu'il ne reste plus qu'à interconnecter, en logique programmable il faut traduire ces équations en informations de programmation du circuit.

Si une telle opération reste faisable à la main, mais avec tout de même un grand risque d'erreur, pour quelques PAL très simples, elle est tout à fait irréalisable pour les circuits plus complexes que sont les "gros" PAL, les EPLD et surtout les FPGA.

Pour tous ces circuits, le recours à un système de développement est donc indispensable. Nous verrons dans les chapitres qui suivent, quels sont les systèmes utilisables pour chaque famille de circuit. Vous pouvez cependant déjà noter qu'ils admettent en général tous en entrée les équations logiques à réaliser et qu'ils savent tous en déduire les informations de programmation adéquates. Ils disposent aussi presque tous de fonctionnalités plus puissantes telles que l'entrée directe de tables de vérité, voire des schémas logiques classiques qu'ils se chargent alors eux-mêmes de traduire en équations. Cette dernière approche est très agréable lorsque l'on veut intégrer une logique déjà conçue, ou bien encore pour faciliter l'utilisation de circuits logiques programmables par des personnes habituées à concevoir des schémas avec des circuits logiques classiques.

Même s'il existe un certain nombre de systèmes de développement tournant sur gros calculateurs ou stations de travail ; systèmes qui peuvent d'ailleurs être intégrés à des logiciels de saisie de schéma ; quasiment tous les produits actuels sont proposés sur compatibles PC à des prix qui restent à la portée de petites entreprises ou de lycées et collèges techniques.

Dans tous les cas, ces systèmes de développement sont capables de produire un fichier JEDEC (voir précédemment) qu'il est alors possible de transmettre à n'importe quel programmateur normalisé. Toutefois, on rencontre de plus en plus souvent l'association système de développement/carte de programmation, le tout fonctionnant sur compatible PC, bien sûr. On passe ainsi directement de la phase de conception de circuit à la production du boîtier proprement dit, ce qui est simple, efficace et performant.

Ceci est d'autant plus vrai pour les circuits tels que les ispLSI, certains EPLD et les FPGA qui, étant programmables en circuit, n'ont bien souvent besoin que d'un câble et d'une circuiterie élémentaire pour pouvoir être programmés directement à partir d'un port série ou parallèle standard du PC utilisé.

Tout ceci étant vu, entrons maintenant dans le vif du sujet avec les premiers circuits programmables de notre classification : les PROM.

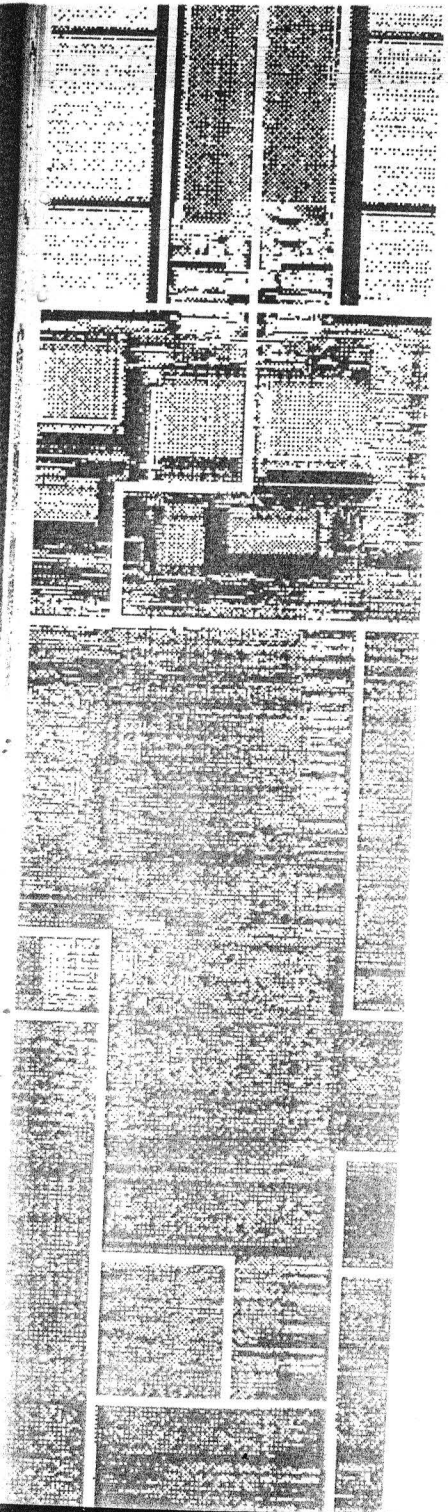
CIRCUITS LOGIQUES PROGRAMMABLES

CHAPITRE 2

Les mémoires mortes programmables ou PROM

EEPROM, EPROM, E2 PROM, FLASH, PROM, UVPROM

- Les PROM bipolaires à fusibles 25
- Les UVPROM ou PROM effaçables aux ultra-violets 33
- Les One Time PROM ou OTP . 53
- Les PROM effaçables électriquement ou EEPROM .. 54
- Les mémoires flash ou flash EPROM 59
- Utilisation des mémoires comme circuits logiques 70



Comme nous l'avons vu au chapitre 1, les mémoires mortes programmables par l'utilisateur ou PROM sont subdivisées en un certain nombre de familles aux possibilités et caractéristiques parfois très diverses. Nous allons donc tout simplement passer en revue ces diverses familles. Nous terminerons ensuite ce chapitre par quelques exemples d'utilisation de ces mémoires en tant que circuits logiques à part entière, ce qui assurera une transition en douceur vers les PAL, objets du chapitre suivant.

LES PROM BIPOLAIRES À FUSIBLES

Si, lors de la classification que nous venons de voir, nous n'avons pas fait intervenir de notions technologiques, ce ne peut plus être le cas ici. En effet, vous allez très vite constater que les grandes familles de PROM découlent en fait des grandes familles technologiques de fabrication.

Les premières PROM qui sont arrivées sur le marché, il y a de cela plus de vingt ans, ont été les PROM bipolaires à fusibles. Ces mémoires étaient en technologie bipolaire car c'était la seule technologie parfaitement maîtrisée à l'époque pour réaliser sur site des mémoires mortes. Les fusibles, quant à eux, étaient le seul moyen disponible à l'époque pour assurer un maintien permanent de l'information placée dans la mémoire.

Cette technologie, même si elle ne représente plus aujourd'hui l'essentiel des ventes, ne doit pas être regardée avec dédain. C'est en effet la seule qui permette, pour le moment, de proposer des PROM aux temps d'accès très courts, aussi faibles que 20 ns par exemple. Les PROM à fusibles ont donc encore quelques beaux jours devant elles dans tous les domaines de la logique ultra rapide.

PRINCIPE D'UNE PROM À FUSIBLES

La zone mémoire proprement dite d'une telle PROM est une matrice de fils croisés, aux intersections desquels sont placées des diodes en série avec des fusibles, comme schématisé figure 2.1. Ces diodes et ces fusibles établissent donc des liaisons à chaque intersection, intersections qui correspondent chacune à un bit de mémoire.

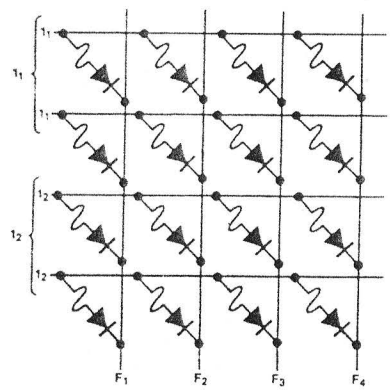


Figure 2.1 :
Principe du plan mémoire d'une PROM à fusibles.

En fait, cette approche, utilisée dans les années 60 pour les matrices de diodes programmables, est un peu améliorée dans les PROM à fusibles. Les lignes horizontales de la matrice sont en effet alimentées par les sorties d'une logique destinée à faciliter l'adressage de la mémoire. Une PROM à fusibles a donc en fait l'allure présentée en figure 2.2. Pour parler en termes de circuits logiques, on peut dire que c'est une structure à ET fixes (les circuits de décodage d'adresse) et à OU programmables (les données placées dans la mémoire).

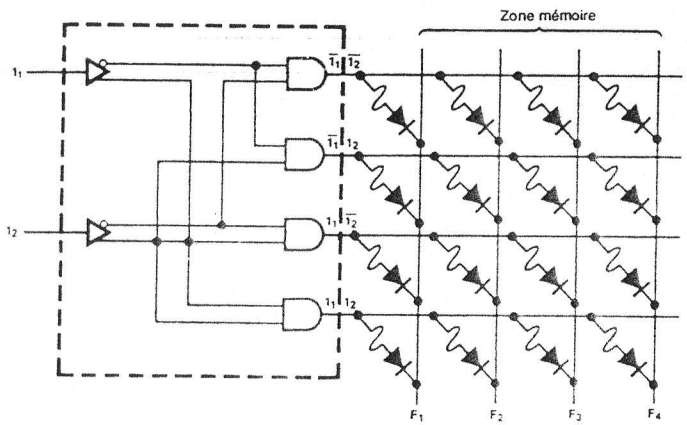


Figure 2.2 :
Schéma de principe d'une PROM à fusibles ; ici une 2 mots de 4 bits.

Lorsque la mémoire vous est livrée, elle est vierge (en principe !) et tous les fusibles sont intacts. Elle ne contient alors que des 1 (par exemple). La programmation va consister à faire sauter les fusibles aux emplacements où vous souhaitez mémoriser des 0. Pratiquement, le programmateur va devoir pour cela adresser votre mémoire, prendre en compte la donnée que vous souhaitez placer à cette adresse et générer un chronogramme particulier, utilisant une haute tension (généralement 12 volts), pour faire sauter le ou les fusibles nécessaires.

Si les chronogrammes et la tension préconisés par le fabricant de la mémoire sont bien respectés, la destruction du fusible est totale et le contenu de la mémoire est définitivement figé, le rétablissement du fusible étant à priori impossible. Le contenu d'une telle mémoire est donc aussi stable dans le temps que celui des ROM programmées par masque à la fabrication puisqu'il y a, là aussi, rupture ou maintient physique de connexion.

Deux remarques importantes sont à faire à ce sujet. D'une part, il est possible de retoucher en partie le contenu d'une telle PROM si vous avez fait un erreur. En effet, si vous avez oublié de faire sauter des fusibles, vous pouvez toujours remettre la mémoire sur le programmateur et les faire sauter quand bon vous semble. Rien n'impose en effet de programmer toute la mémoire d'un seul coup. Bien sûr, si vous avez fait sauter des fusibles alors qu'il ne fallait pas, aucun retour en arrière n'est possible (d'où l'expression retoucher en partie que nous avons employée précédemment).

Si le programmateur utilisé n'est pas homologué par le fabricant de mémoire, les chronogrammes qu'il génère peuvent ne pas être tout à fait corrects (cas de certains montages peu coûteux ou mal bricolés) et la destruction des fusibles peut n'être que partielle. Certains d'entre eux peuvent alors se rétablir, plus ou moins bien, avec le temps. Les niveaux logiques correspondants fournis par la mémoire peuvent devenir impropres à toute utilisation sérieuse. Ceci n'est pas du à un défaut de la technologie des PROM fusibles, mais à l'utilisation d'un mauvais programmateur.

Du fait de leur technologie qui est bipolaire comme nous venons de le dire (à transistors *classiques* si vous préférez, par opposition aux circuits MOS ou CMOS) ; les PROM à fusibles consomment pas mal de courant, ce qui en limite la taille en raison des dissipations de puissance permises par les boîtiers standards. À l'heure actuelle, on trouve de nombreuses configurations allant de 32 mots de 8 bits à 16 K mots de 8 bits.

La figure 2.3 ci-jointe donne à titre d'exemple la liste des PROM bipolaires à fusibles proposées par AMD et Philips (Signetics) à la date de rédaction de cet ouvrage.

Lorsque les UVPROM en étaient encore à leur débuts, les PROM bipolaires étaient souvent utilisées même dans des applications où leur

AMD	Philips	Taille
27S19	–	32 x 8
–	82S23	32 x 8
–	82S123	32 x 8
27S21	–	256 x 4
–	82S126	256 x 4
–	82S129	256 x 4
27S13	82S130	512 x 4
–	82S131	512 x 4
27S29	–	512 x 8
–	82S135	512 x 8
27S31	–	512 x 8
27S33	–	1024 x 4
–	82S137	1024 x 4
27S181	82S181	1024 x 8
27S185	82S185	2048 x 4
27S191	82S191	2048 x 8
27S41	–	4096 x 4
–	82HS195	4096 x 4
27S43	–	4096 x 8
–	82HS321	4096 x 8
27S49	–	8192 x 8
–	82HS641	8192 x 8
27S51	–	16384 x 8

Figure 2.3 :
Les PROM bipolaires proposées par AMD et Philips.

extrême rapidité n'était pas nécessaire. En effet, c'était la seule technologie capable de produire des boîtiers avec des capacités aussi importantes (cela fait sourire aujourd'hui bien sûr !). Depuis, et comme nous le verrons plus loin, les fortes capacités mémoire sont l'apanage des mémoires CMOS de type UVPROM, EEPROM ou FLASH.

Les PROM bipolaires ne sont donc plus utilisées que lorsque leur temps d'accès très court est vraiment nécessaire. En effet, sauf dans de rares cas particuliers, on ne sait pas faire d'EEPROM ou d'UVPROM de temps d'accès aussi faible que celui des meilleures PROM bipolaires à fusibles. Ces dernières *descendent* en effet jusqu'à 20 ns de temps d'accès contre 100 ns pour les plus rapides des mémoires FLASH actuelles par exemple.

CARACTÉRISTIQUES PRINCIPALES

La durée de rétention de l'information dans une PROM à fusible correctement programmée est évidemment infinie.

Les PROM à fusibles ne se distinguent pas particulièrement des autres mémoires tant qu'elles sont utilisées en lecture. Comme le montre la figure 2.4, on y rencontre donc un certain nombre de lignes d'adresses dépendant de la capacité de la mémoire, un certain nombre de lignes de données, dépendant de l'organisation de la mémoire et une ou plusieurs lignes de validation appelées *chip select* ou *chip enable*, actives au niveau haut ou bas. Deux pattes d'alimentation, une masse et un + 5 volts, complètent ce tableau car toutes ces mémoires sont compatibles TTL et s'alimentent donc sous 5 volts.

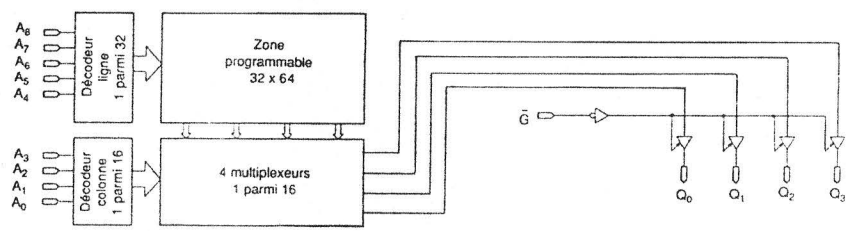
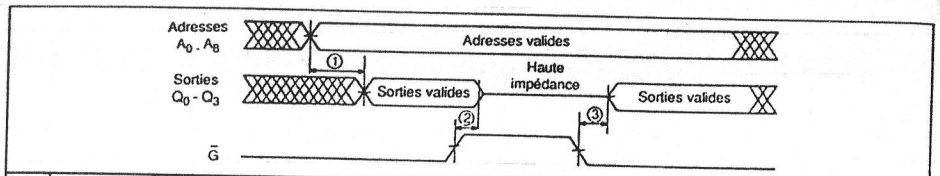


Figure 2.4 :
Synoptique interne d'une PROM à fusibles.

La figure 2.5 rappelle le chronogramme de lecture typique d'une telle mémoire sur lequel on a indiqué à titre d'exemple les divers temps significatifs dans le cas d'une 27S13 d'AMD (PROM de 512 mots de 4 bits). Le paramètre fondamental est évidemment le temps d'accès, toujours très faible pour les PROM bipolaires. La figure 2.3 vous permet de le vérifier sans mal pour ce qui est des références actuellement commercialisées.



No.	Parameter Symbol	Parameter Description	"A" Version COM'L Min.	"A" Version COM'L Max.	"A" Version MIL Min.	"A" Version MIL Max.	Standard Version COM'L Min.	Standard Version COM'L Max.	Standard Version MIL Min.	Standard Version MIL Max.	Unit
1	TAVQV	Address Valid to Output Valid Access Time		30		40		50		60	ns
2	TGVQZ	Delay from Output Enable Valid to Output Hi-Z		20		25		25		30	ns
3	TGVQV	Delay from Output Enable Valid to Output Valid		20		25		25		30	ns

Figure 2.5 : Chronogramme de lecture d'une PROM à fusibles; ici une 27S13 d'AMD.

PROGRAMMATION DES PROM À FUSIBLES

Les PROM à fusibles sont évidemment programmables sur tout appareil normalisé, pour peu que celui-ci comporte dans sa liste de produits acceptés la référence exacte à programmer. En effet, bien que les diverses PROM présentes sur le marché utilisent la même technologie de base et soient toutes compatibles broche à broche en lecture, elles ne sont pas interchangeables en programmation. Dans le meilleur des cas on risque une programmation peu fiable ; dans le pire des cas, la mémoire peut être détruite.

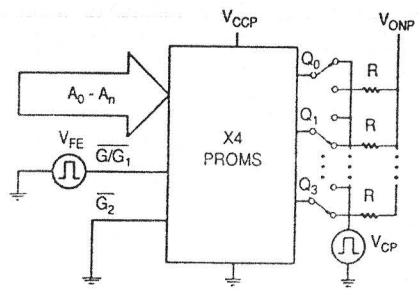
Il faut savoir que, compte tenu de la technologie de fabrication qui interdit tout test de la zone programmable elle-même, les fabricants ne peuvent garantir que leurs mémoires seront à 100 % programmables. De ce fait, ils échangent gratuitement tous les boîtiers qui s'avèrent défectueux en programmation à la seule condition que le programmateur utilisé soit un modèle agréé par leurs soins. Tous les programmateurs de grandes marques sont généralement dans ce cas, mais il vaut mieux, parfois, s'en assurer si l'on a de la production de série à faire.

Compte tenu de leur structure interne relativement simple, les PROM à fusibles peuvent toutefois être programmées avec des moyens rudimentaires. Si une telle approche n'est pas acceptable en production de série, il est tout à fait possible de la mettre en œuvre dans le cadre

d'une étude particulière, ou bien encore à titre d'exercice dans une école ou lycée technique suite à un cours sur les mémoires par exemple.

La seule opération un peu délicate est de se procurer auprès du fabricant les spécifications de programmation qui, bien que théoriquement non confidentielles, ne sont pas toujours faciles à obtenir.

La figure 2.6 associée à la procédure décrite ci-après permet, à titre d'exemple, de comprendre la méthode de programmation utilisée pour les mémoires AMD à mots de 4 bits et sans registres internes de sortie.



Référence	Patte de validation de fusible
Am27S20/21	$\overline{G_1}$
Am27S12/13	$\overline{G}$
Am27S32/33	$\overline{G_1}$
Am27S184/185	$\overline{G}$
Am27S40/41	$\overline{G_1}$

Figure 2.6 : Principe de programmation d'une PROM à fusibles.

Comme on le voit, les lignes d'adresses gardent leur fonction première qui est d'adresser la cellule mémoire à programmer. Les lignes de données, unidirectionnelles en mode lecture, deviennent ici des entrées et vont permettre l'écriture de la donnée à l'adresse sélectionnée. L'action sur les lignes GX de validation du boîtier mais aussi l'augmentation de la tension d'alimentation vont déclencher la procédure de programmation proprement dite qui se passe exactement de la façon suivante :

- Alimentez le circuit sous sa tension nominale de fonctionnement de 5 volts.
- Appliquez l'adresse de la cellule à programmer sur les lignes A_0 à AX. En aucun cas ne laissez une ligne d'adresse *en l'air*. Toutes doivent être ramenées à un vrai niveau logique 1 ou 0.

- Placez la ou les lignes GX au niveau bas et les lignes GX barre au niveau haut pour désactiver la mémoire. Comme pour les adresses, ces actions se font avec de vrais niveaux logiques en ne laissant aucune ligne *en l'air*.
- Appliquez la tension de programmation sur une sortie.
- Appliquez la tension de validation des fusibles pendant la durée préconisée selon le type de PROM qui est de 50 µs lors de la première tentative de programmation.
- Enlevez la tension de programmation de la sortie concernée.
- Validez la mémoire en lecture par mise aux niveaux corrects de ses diverses pattes GX et GX barre et vérifiez que la sortie programmée est bien au niveau voulu. Si ce n'est pas le cas, la procédure de programmation doit être répétée pour ce fusible avec des durées d'impulsions de 5 ms cette fois-ci.
- Lorsque toute la mémoire a été programmée, une vérification de son contenu doit être effectuée avec des tensions d'alimentation de 4,4 volts et de 5,7 volts.

La figure 2.7 précise l'enchaînement de ces opérations sous forme de chronogramme des signaux utilisés alors que la figure 2.8 précise les valeurs numériques des différents paramètres.

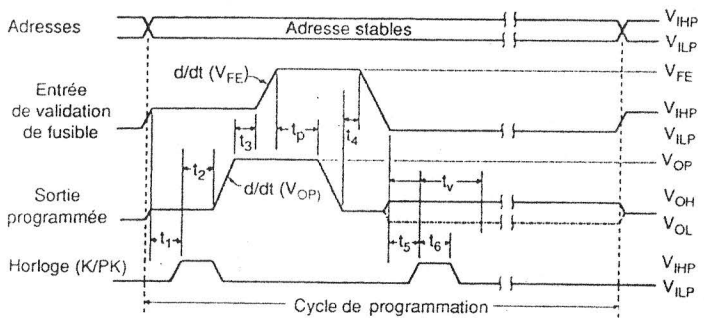


Figure 2.7 : Chronogrammes de programmation d'une PROM à fusibles.

Parameter Symbol	Parameter Description	Min.	Typ.	Max.	Units
V_{IHH}	Control Pin Extra High Level @ 10 – 40 mA	14.5	15	15.5	Volts
V_{FE}	Fusing Enable Voltage @ 10 – 40 mA	14.5	15	15.5	Volts
V_{OP}	Program Voltage @ 15 – 200 mA	19.5	20	20.5	Volts
V_{IHP}	Input High Level During Programming and Verify	2.4	5	5.5	Volts
V_{ILP}	Input Low Level During Programming and Verify	0.0	0.3	0.5	Volts
V_{CCP}	V_{CC} During Programming @ I_{CC} = 50 – 200 mA	5	5.2	5.5	Volts
dV_{OP}/dt	Rate of Output Voltage Change	20		250	V/µs
dV_{FE}/dt	Rate of Fusing Enable Voltage Change	20		1000	V/µs
t_p	Fusing Time First Attempt	10	50	100	µs
	Fusing Time Subsequent Attempts	1	5	10	ms
$t_1 - t_6$	Delays Between Various Level Changes	100		1500	ns
t_V	Period During which Output is Sensed for V_{Blown} Level		500		ns
V_{ONP}	Pull-Up Voltage On Outputs Not Being Programmed	V_{CCP} – 0.3	V_{CCP}	V_{CCP} + 0.3	Volts
R	Pull-Up Resistor On Outputs Not Being Programmed	0.2	2	5.1	KΩ

Figure 2.8 : Paramètres de programmation d'une PROM à fusibles.

On le voit, une telle procédure n'a rien de particulièrement complexe et peut facilement être mise en œuvre sur une plaque de câblage rapide avec quelques résistances, monostables et interrupteurs pour peu bien sûr que l'on n'ait que quelques bits à programmer.

LES UVPROM OU PROM EFFAÇABLES AUX ULTRA-VIOLETS

Les UVPROM existent depuis plus de 15 ans, mais les circuits actuellement commercialisés n'ont plus aucun rapport avec leurs ancêtres au plan des performances. Notre but n'étant pas de faire un manuel d'histoire mais un livre pratique, nous ne parlerons donc pas des *ancêtres* que sont les 1702, 2708 et 2716 tri-tension.

Une fois n'est pas coutume, les fabricants de tous horizons se sont entendus lors de la création des divers circuits de cette famille de mémoires UVPROM et ont tous adopté des brochages sinon identiques, du moins parfaitement compatibles, et des références tout aussi compatibles et logiques dans leur formulation.

Il faut savoir en outre que la majorité des mémoires de ce type est organisée en mots de 8 bits ou, pour les plus récentes d'entre elles, en mots de 16 bits.

Ceci étant vu, la référence d'une mémoire UVPROM donne directement sa capacité. Ainsi, une 2764 est-elle une 64 Kbits, c'est-à-dire une 8 Kmots de 8 bits alors qu'une 27256 est un 256 Kbits, c'est-à-dire une 64 Kmots de 8 bits. Bien sûr, si la mémoire est vendue par Texas, elle s'appellera TMS 27256, alors que chez AMD ce sera Am 27256, mais on retrouvera toujours dans la *racine* de la référence cette indication de taille. C'est déjà un point très important.

Au-dessus de 1 mégabits, c'est un peu plus délicat car les mémoires pouvant être organisées en mots de 8 ou 16 bits, il a fallu faire quelques entorses à cette belle règle. Ainsi, une 27010 est-elle une 128 Kmots de 8 bits, alors qu'une 271024 est une 64 Kmots mais de 16 bits cette fois.

PRINCIPE DES UVPROM

Nous n'allons pas rentrer dans le détail technologique de fabrication des UVPROM, car cela nécessite des connaissances poussées en physique du solide et des semiconducteurs. Nous nous limiterons donc à une explication simple mais néanmoins correcte pour comprendre le processus de fonctionnement.

Comme le montre le schéma de principe de la figure 2.9, chaque cellule mémoire élémentaire est constituée d'un transistor MOS dont la

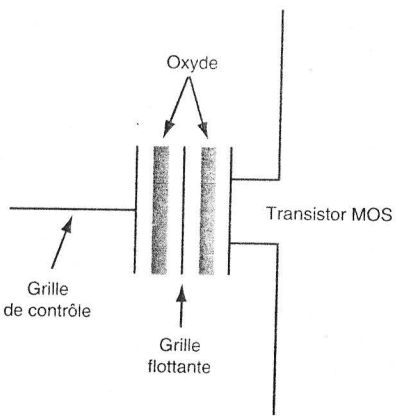


Figure 2.9 :
La cellule mémoire élémentaire d'une UVPROM.

grille de commande est totalement isolée dans une couche d'oxyde. Par application d'une tension suffisamment élevée, qui est appelée tension de programmation, on créé des électrons chauds ou électrons ayant une énergie suffisante pour de passer au travers de cet isolant. Ces charges s'accumulent alors sur cette grille isolée où elles se trouvent piégées. La cellule mémoire est programmée.

Comme la qualité de l'isolant est excellente, la durée de rétention des charges atteint au minimum 10 ans à 70 °C, c'est-à-dire jusqu'à 100 fois plus à 25 °C, ce qui est normalement plus que suffisant en utilisation normale.

Si maintenant on veut effacer la mémoire, on expose la puce à un rayonnement ultraviolet. Les photons, ou particules d'énergie lumineuse, communiquent alors leur énergie aux électrons et leur font franchir la barrière isolante en sens inverse. La grille perd alors ses charges et la mémoire redevient vierge.

Comme il faut tout de même une certaine quantité d'énergie pour le franchissement de cette barrière, le processus n'est pas immédiat et demande une exposition de durée relativement longue (plusieurs minutes à plusieurs dizaines de minutes selon la puissance de la source).

Théoriquement, un tel procédé est réversible à l'infini. En pratique, des phénomènes complexes viennent s'ajouter à ce que nous avons exposé et conduisent à de possibles problèmes au delà de 1000 à 10000 cycles de programmation et effacement. C'est heureusement bien au-delà des besoins réels des utilisateurs.

Ce mode de fonctionnement appelle toutefois quelques remarques que voici. En effet, ces mémoires sont sensibles aux ultraviolets et, s'il ne faut que 10 à 20 minutes d'exposition à une source d'UV suffisamment intense et de bonne longueur d'onde pour les effacer, des expériences ont montré que l'exposition pendant 3 ans à un éclairage fluorescent (néons ordinaires) dégradait fortement leur contenu, tandis qu'il suffisait d'une semaine d'exposition directe au soleil pour arriver au même résultat. Moralité : il faut toujours recouvrir la fenêtre d'une UVPROM d'un adhésif opaque pour ne pas avoir d'ennuis, mais point n'est besoin d'utiliser pour cela certaines étiquettes dites spéciales vendues fort cher par des fournisseurs de composants, un simple morceau de ruban adhésif noir suffit.

La haute tension de programmation de ces mémoires, évoquée précédemment, ne doit pas vous inquiéter ; en effet alors qu'elle était de 25 volts sur les premiers boîtiers commercialisés (2716 et premières 2732), elle est tombée à 21 puis à 12,5 volts sur les mémoires les plus récentes. Cette baisse a d'ailleurs conduit certaines personnes mal documentées à *carboniser* de nombreuses mémoires sur des programmateurs qui ne faisaient pas la différence entre des versions à programmation sous 21 volts et des versions à programmation sous 12,5 volts des mêmes mémoires. En effet, lors de l'évolution technologique ayant permis cette réduction de tension de programmation, des mémoires de même taille mais se programmant soit sous 12,5 volts, soit sous 21 volts ont cohabité sur le marché. Comme les variantes de références n'étaient pas toujours évidentes, on imagine sans mal les possibles confusions.

Du fait de la technique de programmation, il est évidemment possible de programmer n'importe quel octet à n'importe quel moment, aucune obligation n'existe à ce sujet. Par contre, il est évident que l'effacement agit sur toute la mémoire ; on voit mal comment il pourrait en être autrement puisque toute la puce est exposée aux ultra-violets.

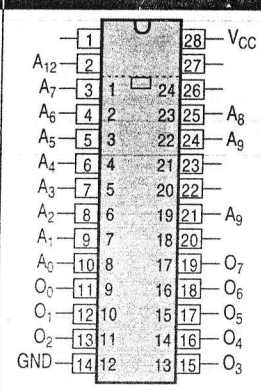
CARACTÉRISTIQUES PRINCIPALES

Nous avons vu précédemment que les références indiquaient directement la taille de la mémoire. À l'heure actuelle, ces références s'étagent de la 2764 ou 8 Kmots de 8 bits à la 274096 ou 256 Kmots de 16 bits.

En fait, on trouve encore sur le marché des 2732 et même parfois des 2716 qui sont respectivement des 4 Kmots de 8 bits et des 2 Kmots de 8 bits. Même si ces mémoires peuvent être utilisées dans des *bidouilles* de laboratoire ou d'amateur, leur utilisation est déconseillée pour toute réalisation nouvelle. En effet, elles sont en voie de disparition au profit des versions à plus forte capacité qui coûtent moins cher (et oui, une 2764 qui est une double 2732 coûte moins cher qu'une 2732 !).

En outre, le brochage normalisé auquel nous faisions allusion, devait faire appel à deux tailles de boîtiers : les boîtiers 24 pattes qui avaient été choisis pour les 2708, 2716 et 2732 car ils disposaient alors d'assez de pattes pour faire passer les lignes d'adresses nécessaires, et les boîtiers à 28 pattes utilisés à partir des 2764. Bien sûr, comme le montre la figure 2.10, les choix de brochages étaient particulièrement astucieux et permettaient de monter des mémoires 24 pattes dans des supports prévus pour évoluer ensuite vers les 28 pattes.

27C512 27512	27C256 27256	27C64 2764	27C32 2732	27C16 2716
A_{15}	V_{PP}	V_{PP}		
A_{12}	A_{12}	A_{12}		
A_7	A_7	A_7	A_7	A_7
A_6	A_6	A_6	A_6	A_6
A_5	A_5	A_5	A_5	A_5
A_4	A_4	A_4	A_4	A_4
A_3	A_3	A_3	A_3	A_3
A_2	A_2	A_2	A_2	A_2
A_1	A_1	A_1	A_1	A_1
A_0	A_0	A_0	A_0	A_0
O_0	O_0	O_0	O_0	O_0
O_1	O_1	O_1	O_1	O_1
O_2	O_2	O_2	O_2	O_2
Masse	Masse	Masse	Masse	Masse



27C16 2716	27C32 2732	27C64 2764	27C256 27256	27C512 275121
		V_{CC} / $\overline{PGM}$	V_{CC}	V_{CC}
			A_{14}	A_{14}
V_{CC}	V_{CC}	NC	A_{13}	A_{13}
A_8	A_8	A_8	A_8	A_8
A_9	A_9	A_9	A_9	A_9
V_{PP}	$\overline{A_{11}}$	$\overline{A_{11}}$	$\overline{A_{11}}$	$\overline{A_{11}}$
$\overline{OE}$	$\overline{OE}/V_{PP}$	$\overline{OE}$	$\overline{OE}$	$\overline{OE}/V_{PP}$
A_{10}	A_{10}	A_{10}	$\overline{A_{10}}$	A_{10}
$\overline{CE}/PGM$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}/PGM$	$\overline{CE}$
O_7	O_7	O_7	O_7	O_7
O_6	O_6	O_6	O_6	O_6
O_5	O_5	O_5	O_5	O_5
O_4	O_4	O_4	O_4	O_4
O_3	O_3	O_3	O_3	O_3

Figure 2.10 :
Le brochage évolutif 24-28 pattes adopté à l'origine des UVPROM.

Ceci n'est plus de mise aujourd'hui et tout produit industriel digne de ce nom doit être conçu au minimum avec un support 28 pattes qui recevra donc au minimum une 2764, même si une toute petite partie de sa capacité seulement est utilisée. C'est la rançon du progrès !

Si des capacités mémoires très importantes doivent être envisagées sur l'application, ce support pourra avantageusement être prévu à 32 pattes puisque, comme nous allons le voir dans un instant, c'est cette taille qui a été choisie pour les boîtiers à très forte capacité organisés en mots de 8 bits. Comme à l'époque du 24/28 pattes, on assiste bien évidemment à une compatibilité ascendante 28/32 pattes.

Ces mémoires UVPROM existent encore actuellement en deux technologies différentes : la technologie la plus ancienne ou NMOS qui présente l'inconvénient de consommer un courant relativement important et de conduire pourtant à des temps d'accès assez importants

et la technologie plus récente CMOS rapide qui permet, tout en consommant beaucoup moins, de proposer des temps d'accès plus faibles et même beaucoup plus faibles.

Les circuits NMOS sont toutefois en voie de disparition au profit des versions CMOS, plus performantes, et leur utilisation est donc fortement déconseillée pour toute réalisation nouvelle.

Ceci étant précisé, et comme nous l'avons dit précédemment, ces mémoires sont pour la plupart organisées en mots de 8 bits, ce qui leur a valu l'appellation *bytewide* (littéralement large d'un octet) lors de leurs mise sur le marché. Ceci est resté vrai jusqu'à la commercialisation des mémoires de 1 mégabits qui a coïncidé avec le développement fulgurant des microprocesseurs 16 et 32 bits. Ces dernières mémoires ont donc été proposées en versions 8 et 16 bits.

Pour préciser tout cela, la figure 2.11, extraite du data book AMD donne un aperçu réaliste des diverses références disponibles à la date

Figure 2.11 :
Les différentes UVPROM disponibles chez AMD à la date de rédaction de cet ouvrage.

Référence	Capacité	Temps d'accès (ns)	Gamme de température	Boîtier	Nombre de pattes (DIP/PLCC TSOP)	Tension d'alimentation
Am27C64-55	8K x 8	55	C	D, L	28/32	5 V ± 5%
Am27C64-70	8K x 8	70	C	D, L	28/32	5 V ± 10%
Am27C64-90	8K x 8	90	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C64-120	8K x 8	120	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C64-150	8K x 8	150	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C64-200	8K x 8	200	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C64-255	8K x 8	250	C, I	D, L, P, J	28/32	5 V ± 5%
Am27C128-55	16K x 8	55	C	D, L	28/32	5 V ± 5%
Am27C128-70	16K x 8	70	C	D, L	28/32	5 V ± 10%
Am27C128-90	16K x 8	90	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C128-120	16K x 8	120	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C128-150	16K x 8	150	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C128-200	16K x 8	200	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C128-255	16K x 8	250	C, I	D, L, P, J	28/32	5 V ± 5%
Am27H256-35	32K x 8	35	C	D, L	28/32	5 V ± 10%
Am27H256-35V05	32K x 8	35	C	D, L	28/32	5 V ± 5%
Am27H256-45	32K x 8	45	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27H256-55	32K x 8	55	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27H256-70	32K x 8	70	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C256-55	32K x 8	55	C	D, L	28/32	5 V ± 5%
Am27C256-70	32K x 8	70	C	D, L	28/32	5 V ± 10%
Am27C256-90	32K x 8	90	C, I, E, M	D, L, P, J, E	28/32	5 V ± 10%
Am27C256-120	32K x 8	120	C, I, E, M	D, L, P, J, E	28/32	5 V ± 10%
Am27C256-150	32K x 8	150	C, I, E, M	E	28/32	5 V ± 10%
Am27C256-200	32K x 8	200	C, I, E, M	D, L, P, J, E	28/32	5 V ± 10%
Am27C256-255	32K x 8	250	C, I	D, L, P, J	28/32	5 V ± 5%

Référence	Capacité	Temps d'acces (ns)	Gamme de température	Boitier	Nombre de pattes (DIP PLCC TSOP)	Tension d'alimentation
Am27C512-75	**64K x 8**	**70**	**C**	**D, L**	**28/32**	**5 V ± 5%**
Am27C512-90	**64K x 8**	**90**	**C, I, E, M**	**D, L**	**28/32**	**5 V ± 10%**
Am27C512-120	**64K x 8**	**120**	**C, I, E, M**	**D, L**	**28/32**	**5 V ± 10%**
Am27C512-150	**64K x 8**	**150**	**C, I, E, M**	**D, L, P, J**	**28/32**	**5 V ± 10%**
Am27C512-200	**64K x 8**	**200**	**C, I, E, M**	**D, L, P, J**	**28/32**	**5 V ± 10%**
Am27C512-255	**64K x 8**	**250**	**C, I, E, M**	**D, L, P, J**	**28/32**	**5 V ± 5%**
Am27H010-45	128K x 8	45	C	D, L	32/32	5 V ± 10%
Am27H010-45V05	128K x 8	45	C	D, L	32/32	5 V ± 5%
Am27H010-55	128K x 8	55	C, I, E, M	D, L, P, J	32/32	5 V ± 10%
Am27H010-70	128K x 8	70	C, I, E, M	D, L, P, J	32/32	5 V ± 10%
Am27H010-90	128K x 8	90	C, I, E, M	D, L, P, J	32/32	5 V ± 10%
Am27H010-90V05	128K x 8	90	C, I, E, M	D, L, P, J	32/32	5 V ± 5%
Am27C010-105	128K x 8	100	C	D, L	32/32	5 V ± 5%
Am27C010-120	128K x 8	120	C, I	D, L, P, J, E	32/32	5 V ± 10%
Am27C010-150	128K x 8	150	C, I, E, M	D, L, P, J, E	32/32	5 V ± 10%
Am27C010-200	128K x 8	200	C, I, E, M	D, L, P, J, E	32/32	5 V ± 10%
Am27C010-255	128K x 8	250	C, I	D, L, P, J, E	32/32	5 V ± 5%
Am27LV010-120	128K x 8	120	C, I, E	D, L	32	3.3 V ± 10%
Am27LV010-150	128K x 8	150	C, I, E, M	D, L, J, E	32	3.3 V ± 10%
Am27LV010-200	128K x 8	200	C, I, E, M	D, L, J, E	32	3.3 V ± 10%
Am27LV010-250	128K x 8	250	C, I, E, M	D, L, J, E	32	3.3 V ± 10%
Am27LV010-300	128K x 8	300	C, I, E, M	D, L, J, E	32	3.3 V ± 10%
Am27LV010B-150	128K x 8	150	C, I, E	D, L, J, E	32	2.7 V – 3.6 V
Am27LV010B-200	128K x 8	200	C, I, E	D, L, J, E	32	2.7 V – 3.6 V
Am27LV010B-250	128K x 8	250	C, I, E, M	D, L, J, E	32	2.7 V – 3.6 V
Am27LV010B-300	128K x 8	300	C, I, E, M	D, L, J, E	32	2.7 V – 3.6 V
Am27C1024-85	64K x 16	85	C	D	40	5 V ± 5%
Am27C1024-90	64K x 16	90	C, I	D, L	40/44	5 V ± 10%
Am27C1024-120	64K x 16	120	C, I, E, M	D, L	40/44	5 V ± 10%
Am27C1024-150	64K x 16	150	C, I, E, M	D, L	40/44	5 V ± 10%
Am27C1024-200	64K x 16	200	C, I, E, M	D, L, P, J	40/44	5 V ± 10%
Am27C1024-255	64K x 16	250	C, I	D, L, P, J	40/44	5 V ± 5%
Am27C020-120	256K x 8	120	C, I	D, L	32/32	5 V ± 10%
Am27C020-150	256K x 8	150	C, I, E, M	D, L, P, J*	32/32	5 V ± 10%
Am27C020-200	256K x 8	200	C, I, E, M	D, L, P, J*	32/32	5 V ± 10%
Am27C020-250	256K x 8	250	M	D, L*	32/32	5 V ± 10%
Am27C020-255	256K x 8	250	C, I	D, L, P, J*	32/32	5 V ± 5%
Am27LV020-150	256K x 8	150	C, I, E	D, L, J	32	3.3 V ± 10%
Am27LV020-200	256K x 8	200	C, I, E, M	D, L, J	32	3.3 V ± 10%
Am27LV020-250	256K x 8	250	C, I, E, M	D, L, J	32	3.3 V ± 10%
Am27LV020-300	256K x 8	300	C, I, E, M	D, L, J	32	3.3 V ± 10%
Am27LV020B-200	128K x 8	200	C, I, E	D, L, J	32	2.7 V – 3.6 V
Am27LV020B-250	128K x 8	250	C, I, E, M	D, L, J	32	2.7 V – 3.6 V
Am27LV020B-300	128K x 8	300	C, I, E, M	D, L, J	32	2.7 V – 3.6 V
Am27C2048-105*	128K x 16	100	C	D, L	40/44	5 V ± 5%
Am27C2048-120	128K x 16	120	C, I	D, L	40/44	5 V ± 10%
Am27C2048-150	128K x 16	150	C, I, E, M	D, L, P, J	40/44	5 V ± 10%
Am27C2048-200	128K x 16	200	C, I, E, M	D, L, P, J	40/44	5 V ± 10%
Am27C2048-250	128K x 16	250	M	D, L	40/44	5 V ± 10%
Am27C2048-255	128K x 8	250	C, I	D, L, P, J	40/44	5 V ± 5%

et la technologie plus récente CMOS rapide qui permet, tout en consommant beaucoup moins, de proposer des temps d'accès plus faibles et même beaucoup plus faibles.

Les circuits NMOS sont toutefois en voie de disparition au profit des versions CMOS, plus performantes, et leur utilisation est donc fortement déconseillée pour toute réalisation nouvelle.

Ceci étant précisé, et comme nous l'avons dit précédemment, ces mémoires sont pour la plupart organisées en mots de 8 bits, ce qui leur a valu l'appellation *bytewide* (littéralement large d'un octet) lors de leurs mise sur le marché. Ceci est resté vrai jusqu'à la commercialisation des mémoires de 1 mégabits qui a coïncidé avec le développement fulgurant des microprocesseurs 16 et 32 bits. Ces dernières mémoires ont donc été proposées en versions 8 et 16 bits.

Pour préciser tout cela, la figure 2.11, extraite du data book AMD donne un aperçu réaliste des diverses références disponibles à la date

Figure 2.11 :
Les différentes UVPROM disponibles chez AMD à la date de rédaction de cet ouvrage.

Référence	Capacité	Temps d'accès (ns)	Gamme de température	Boîtier	Nombre de pattes (DIP/PLCC TSOP)	Tension d'alimentation
Am27C64-55	8K x 8	55	C	D, L	28/32	5 V ± 5%
Am27C64-70	8K x 8	70	C	D, L	28/32	5 V ± 10%
Am27C64-90	8K x 8	90	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C64-120	8K x 8	120	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C64-150	8K x 8	150	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C64-200	8K x 8	200	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C64-255	8K x 8	250	C, I	D, L, P, J	28/32	5 V ± 5%
Am27C128-55	16K x 8	55	C	D, L	28/32	5 V ± 5%
Am27C128-70	16K x 8	70	C	D, L	28/32	5 V ± 10%
Am27C128-90	16K x 8	90	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C128-120	16K x 8	120	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C128-150	16K x 8	150	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C128-200	16K x 8	200	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C128-255	16K x 8	250	C, I	D, L, P, J	28/32	5 V ± 5%
Am27H256-35	32K x 8	35	C	D, L	28/32	5 V ± 10%
Am27H256-35V05	32K x 8	35	C	D, L	28/32	5 V ± 5%
Am27H256-45	32K x 8	45	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27H256-55	32K x 8	55	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27H256-70	32K x 8	70	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C256-55	32K x 8	55	C	D, L	28/32	5 V ± 5%
Am27C256-70	32K x 8	70	C	D, L	28/32	5 V ± 10%
Am27C256-90	32K x 8	90	C, I, E, M	D, L, P, J, E	28/32	5 V ± 10%
Am27C256-120	32K x 8	120	C, I, E, M	D, L, P, J, E	28/32	5 V ± 10%
Am27C256-150	32K x 8	150	C, I, E, M	E	28/32	5 V ± 10%
Am27C256-200	32K x 8	200	C, I, E, M	D, L, P, J, E	28/32	5 V ± 10%
Am27C256-255	32K x 8	250	C, I	D, L, P, J	28/32	5 V ± 5%

Référence	Capacité	Temps d'acces (ns)	Gamme de température	Boîtier	Nombre de pattes (DIP PLCC TSOP)	Tension d'alimentation
Am27C512-75	64K x 8	70	C	D, L	28/32	5 V ± 5%
Am27C512-90	64K x 8	90	C, I, E, M	D, L	28/32	5 V ± 10%
Am27C512-120	64K x 8	120	C, I, E, M	D, L	28/32	5 V ± 10%
Am27C512-150	64K x 8	150	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C512-200	64K x 8	200	C, I, E, M	D, L, P, J	28/32	5 V ± 10%
Am27C512-255	64K x 8	250	C, I, E, M	D, L, P, J	28/32	5 V ± 5%
Am27H010-45	128K x 8	45	C	D, L	32/32	5 V ± 10%
Am27H010-45V05	128K x 8	45	C	D, L	32/32	5 V ± 5%
Am27H010-55	128K x 8	55	C, I, E, M	D, L, P, J	32/32	5 V ± 10%
Am27H010-70	128K x 8	70	C, I, E, M	D, L, P, J	32/32	5 V ± 10%
Am27H010-90	128K x 8	90	C, I, E, M	D, L, P, J	32/32	5 V ± 10%
Am27H010-90V05	128K x 8	90	C, I, E, M	D, L, P, J	32/32	5 V ± 5%
Am27C010-105	128K x 8	100	C	D, L	32/32	5 V ± 5%
Am27C010-120	128K x 8	120	C, I	D, L, P, J, E	32/32	5 V ± 10%
Am27C010-150	128K x 8	150	C, I, E, M	D, L, P, J, E	32/32	5 V ± 10%
Am27C010-200	128K x 8	200	C, I, E, M	D, L, P, J, E	32/32	5 V ± 10%
Am27C010-255	128K x 8	250	C, I	D, L, P, J, E	32/32	5 V ± 5%
Am27LV010-120	128K x 8	120	C, I, E	D, L	32	3.3 V ± 10%
Am27LV010-150	128K x 8	150	C, I, E, M	D, L, J, E	32	3.3 V ± 10%
Am27LV010-200	128K x 8	200	C, I, E, M	D, L, J, E	32	3.3 V ± 10%
Am27LV010-250	128K x 8	250	C, I, E, M	D, L, J, E	32	3.3 V ± 10%
Am27LV010-300	128K x 8	300	C, I, E, M	D, L, J, E	32	3.3 V ± 10%
Am27LV010B-150	128K x 8	150	C, I, E	D, L, J, E	32	2.7 V – 3.6 V
Am27LV010B-200	128K x 8	200	C, I, E	D, L, J, E	32	2.7 V – 3.6 V
Am27LV010B-250	128K x 8	250	C, I, E, M	D, L, J, E	32	2.7 V – 3.6 V
Am27LV010B-300	128K x 8	300	C, I, E, M	D, L, J, E	32	2.7 V – 3.6 V
Am27C1024-85	64K x 16	85	C	D	40	5 V ± 5%
Am27C1024-90	64K x 16	90	C, I	D, L	40/44	5 V ± 10%
Am27C1024-120	64K x 16	120	C, I, E, M	D, L	40/44	5 V ± 10%
Am27C1024-150	64K x 16	150	C, I, E, M	D, L	40/44	5 V ± 10%
Am27C1024-200	64K x 16	200	C, I, E, M	D, L, P, J	40/44	5 V ± 10%
Am27C1024-255	64K x 16	250	C, I	D, L, P, J	40/44	5 V ± 5%
Am27C020-120	256K x 8	120	C, I	D, L	32/32	5 V ± 10%
Am27C020-150	256K x 8	150	C, I, E, M	D, L, P, J*	32/32	5 V ± 10%
Am27C020-200	256K x 8	200	C, I, E, M	D, L, P, J*	32/32	5 V ± 10%
Am27C020-250	256K x 8	250	M	D, L*	32/32	5 V ± 10%
Am27C020-255	256K x 8	250	C, I	D, L, P, J*	32/32	5 V ± 5%
Am27LV020-150	256K x 8	150	C, I, E	D, L, J	32	3.3 V ± 10%
Am27LV020-200	256K x 8	200	C, I, E, M	D, L, J	32	3.3 V ± 10%
Am27LV020-250	256K x 8	250	C, I, E, M	D, L, J	32	3.3 V ± 10%
Am27LV020-300	256K x 8	300	C, I, E, M	D, L, J	32	3.3 V ± 10%
Am27LV020B-200	128K x 8	200	C, I, E	D, L, J	32	2.7 V – 3.6 V
Am27LV020B-250	128K x 8	250	C, I, E, M	D, L, J	32	2.7 V – 3.6 V
Am27LV020B-300	128K x 8	300	C, I, E, M	D, L, J	32	2.7 V – 3.6 V
Am27C2048-105*	128K x 16	100	C	D, L	40/44	5 V ± 5%
Am27C2048-120	128K x 16	120	C, I	D, L	40/44	5 V ± 10%
Am27C2048-150	128K x 16	150	C, I, E, M	D, L, P, J	40/44	5 V ± 10%
Am27C2048-200	128K x 16	200	C, I, E, M	D, L, P, J	40/44	5 V ± 10%
Am27C2048-250	128K x 16	250	M	D, L	40/44	5 V ± 10%
Am27C2048-255	128K x 8	250	C, I	D, L, P, J	40/44	5 V ± 5%

Référence	Capacité	Temps d'accès (ns)	Gamme de température	Boîtier	Nombre de pattes (DIP/PLCC TSOP)	Tension d'alimentation
Am27C040-120	512K x 8	120	C, I	D, L	32/32	5 V ± 10%
Am27C040-125	512K x 8	120	C, I	D, L	32/32	5 V ± 5%
Am27C040-150	512K x 8	150	C, I, E, M	D, L	32/32	5 V ± 10%
Am27C040-200	512K x 8	200	C, I, E, M	D, L, P, J	32/32	5 V ± 10%
Am27C040-250	512K x 8	250	M	D, L	32/32	5 V ± 10%
Am27C040-255	512K x 8	250	C, I	D, L, P, J	32/32	5 V ± 5%
Am27C400-125	512K x 8/256K x 16	120	C, I	D	40	5 V ± 5%
Am27C400-120	512K x 8/256K x 16	120	C, I	D	40	5 V ± 10%
Am27C400-150	512K x 8/256K x 16	150	C, I	D	40	5 V ± 10%
Am27C400-200	512K x 8/256K x 16	200	C, I	D	40	5 V ± 10%
Am27C400-255	512K x 8/256K x 16	250	C, I	D	40	5 V ± 5%
Am27C4096-125	256K x 16	120	C, I	D, L	40/44	5 V ± 5%
Am27C4096-120	256K x 16	120	C, I	D, L	40/44	5 V ± 10%
Am27C4096-150	256K x 16	150	C, I, E, M	D, L, P, J	40/44	5 V ± 10%
Am27C4096-200	256K x 16	200	C, I, E, M	D, L, P, J	40/44	5 V ± 10%
Am27C4096-250	256K x 16	250	M	D, L	40/44	5 V ± 10%
Am27C4096-255	256K x 16	250	C, I	D, L, P, J	40/44	5 V ± 5%
Am27C080-105*	1 Megabit x 8	100	C, I	D, L	32/32	5 V ± 5%
Am27C080-120*	1 Megabit x 8	120	C, I	D, L	32/32	5 V ± 10%
Am27C080-150*	1 Megabit x 8	150	C, I, E, M	D, L, P, J	32/32	5 V ± 10%
Am27C080-200*	1 Megabit x 8	200	C, I, E, M	D, L, P, J	32/32	5 V ± 10%
Am27C080-250*	1 Megabit x 8	250	M	D, L	32/32	5 V ± 10%
Am27C080-255*	1 Megabit x 8	250	C, I	D, L, P, J	32/32	5 V ± 5%
Am27C800-125*	1 Megabit x 8/512K x 16	120	C, I	D, L	42/44	5 V ± 5%
Am27C800-120*	1 Megabit x 8/512K x 16	120	C, I	D, L	42/44	5 V ± 10%
Am27C800-150*	1 Megabit x 8/512K x 16	150	C, I, E, M	D, L, P, J	42/44	5 V ± 10%
Am27C800-200*	1 Megabit x 8/512K x 16	200	C, I, E, M	D, L, P, J	42/44	5 V ± 10%
Am27C800-250*	1 Megabit x 8/512K x 16	250	M	D, L	42/44	5 V ± 10%
Am27C800-255*	1 Megabit x 8/512K x 16	250	C, I	D, L, P, J	42/44	5 V ± 5%

Figure 2.11 :
Les différentes UVPROM disponibles chez AMD à la date de rédaction de cet ouvrage.

de rédaction de cet ouvrage. Toutes ces mémoires sont en technologie CMOS comme en témoigne la lettre C contenue au milieu de la référence. Les versions LV (pour *Low Voltage*), quant à elles, sont des versions 3,3 volts destinées aux équipements portables.

Cette figure permet de vérifier les généralités que nous venons d'exposer et met aussi en évidence les entorses aux règles d'appellation que nous avons évoquées. Ainsi, si la référence 27C1024 est logique (1024 Kbits) la référence 27C010 l'est moins. Elle a du être créée artificiellement puisque les mémoires de 1 mégabits peuvent être organisées en 128 Kmots de 8 bits ou en 64 Kmots de 16 bits. Il en est évidemment de même pour toutes les mémoires de tailles supérieures pour lesquelles doivent co-exister deux références : une en 8 bits et une en 16 bits.

On y remarque par ailleurs l'apparition des boîtiers DIL 32 pattes pour les mémoires organisées en mots de 8 bits ; boîtiers imposés par le nombre de lignes d'adresses nécessaires en raison des capacités sans cesse grandissantes des mémoires.

Les mémoires organisées en mots de 16 bits sont quant à elles proposées en boîtiers DIL 40 pattes pour la même raison mais aussi parce qu'il faut pouvoir en plus faire passer les 16 lignes de données.

En outre, avec l'augmentation sans cesse croissante de la compacité des appareils électroniques actuels, la majorité de ces mémoires est aussi proposée en boîtiers de types LCC ou assimilés qui adoptent, bien évidemment, un brochage standard eux aussi.

Pour que tout soit bien clair, nous vous proposons trois figures résumant la situation :

- En figure 2.12 est présenté le boîtier DIL 28 pattes classique qui est utilisé de la 2764 à la 27512.

27512	27256	2764		2764	27256	27512
A_{15}	V_{PP}	V_{PP}	1 / 28	V_{CC}	V_{CC}	V_{CC}
A_{12}	A_{12}	A_{12}	2 / 27	$\overline{PGM}$	A_{14}	A_{14}
A_7	A_7	A_7	3 / 26	–	A_{13}	A_{13}
A_6	A_6	A_6	4 / 25	A_8	A_8	A_8
A_5	A_5	A_5	5 / 24	A_9	A_9	A_9
A_4	A_4	A_4	6 / 23	A_{11}	A_{11}	A_{11}
A_3	A_3	A_3	7 / 22	$\overline{OE}$	$\overline{OE}$	$\overline{OE}/V_{PP}$
A_2	A_2	A_2	8 / 21	A_{10}	A_{10}	A_{10}
A_1	A_1	A_1	9 / 20	$\overline{CE}$	$\overline{CE}/\overline{PGM}$	$\overline{CE}$
A_0	A_0	A_0	10 / 19	D_7	D_7	D_7
D_0	D_0	D_0	11 / 18	D_6	D_6	D_6
D_1	D_1	D_1	12 / 17	D_5	D_5	D_5
D_2	D_2	D_2	13 / 16	D_4	D_4	D_4
Masse	Masse	Masse	14 / 15	D_3	D_3	D_3

Figure 2.12 :
Brochage du boîtier DIL 28 pattes normalisé.

- En figure 2.13 est présenté le boîtier DIL 32 pattes qui est utilisé de la 27010 à la 27080 donc pour les mémoires de capacité supérieure à 512 Kbits organisées en mots de 8 bit.

27080	27040	27020	27010		27010	27020	27040	27080
A_{19}	V_{PP}	V_{PP}	V_{PP}	1 32	V_{CC}	V_{CC}	V_{CC}	V_{CC}
A_{16}	A_{16}	A_{16}	A_{16}	2 31	$\overline{PGM}$	$\overline{PGM}$	A_{18}	A_{18}
A_{15}	A_{15}	A_{15}	A_{15}	3 30	–	A_{17}	A_{17}	A_{17}
A_{12}	A_{12}	A_{12}	A_{12}	4 29	A_{14}	A_{14}	A_{14}	A_{14}
A_7	A_7	A_7	A_7	5 28	A_{13}	A_{13}	A_{13}	A_{13}
A_6	A_6	A_6	A_6	6 27	A_8	A_8	A_8	A_8
A_5	A_5	A_5	A_5	7 26	A_9	A_9	A_9	A_9
A_4	A_4	A_4	A_4	8 25	A_{11}	A_{11}	$\overline{OE}$	$\overline{OE}/V_{PP}$
A_3	A_3	A_3	A_3	9 24	$\overline{OE}$	$\overline{OE}$	A_{10}	A_{10}
A_2	A_2	A_2	A_2	10 23	A_{10}	A_{10}	$\overline{CE}/\overline{PGM}$	$\overline{CE}/\overline{PGM}$
A_1	A_1	A_1	A_1	11 22	$\overline{CE}$	$\overline{CE}$	D_7	D_7
A_0	A_0	A_0	A_0	12 21	D_7	D_7	D_6	D_6
D_0	D_0	D_0	D_0	13 20	D_6	D_6	D_5	D_5
D_1	D_1	D_1	D_1	14 19	D_5	D_5	D_4	D_4
D_2	D_2	D_2	D_2	15 18	D_4	D_4	D_3	D_3
Masse	Masse	Masse	Masse	16 17	D_3	D_3		

Figure 2.13 :
Brochage du boîtier DIL 32 pattes normalisé.

- En figure 2.14 est présenté le boîtier DIL 40 pattes qui est utilisé pour les mémoires de la 271024 à la 274096 c'est-à-dire pour les mémoires de capacité supérieure à 512 Kbits et organisées en mots de 16 bits.

Malgré ces différences de tailles de boîtiers imposées par des contraintes physiques de nombre de lignes à faire passer, les broches utilisées se retrouvent quasiment toutes identiques à elles-mêmes et ont les mêmes fonctions comme nous allons le voir maintenant grâce à deux exemples représentatifs.

274096	272048	271024		271024	272048	274096
V_{PP}	V_{PP}	V_{PP}	1 40	V_{CC}	V_{CC}	V_{CC}
$\overline{CE}/\overline{PGM}$	$\overline{CE}$	$\overline{CE}$	2 39	$\overline{PGM}$	$\overline{PGM}$	A_{17}
D_{15}	D_{15}	D_{15}	3 38	–	A_{16}	A_{16}
D_{14}	D_{14}	D_{14}	4 37	A_{15}	A_{15}	A_{15}
D_{13}	D_{13}	D_{13}	5 36	A_{14}	A_{14}	A_{14}
D_{12}	D_{12}	D_{12}	6 35	A_{13}	A_{13}	A_{13}
D_{11}	D_{11}	D_{11}	7 34	A_{12}	A_{12}	A_{12}
D_{10}	D_{10}	D_{10}	8 33	A_{11}	A_{11}	A_{11}
D_9	D_9	D_9	9 32	A_{10}	A_{10}	A_{10}
D_8	D_8	D_8	10 31	A_9	A_9	A_9
Masse	Masse	Masse	11 30	Masse	Masse	Masse
D_7	D_7	D_7	12 29	A_8	A_8	A_8
D_6	D_6	D_6	13 28	A_7	A_7	A_7
D_5	D_5	D_5	14 27	A_6	A_6	A_6
D_4	D_4	D_4	15 26	A_5	A_5	A_5
D_3	D_3	D_3	16 25	A_4	A_4	A_4
D_2	D_2	D_2	17 24	A_3	A_3	A_3
D_1	D_1	D_1	18 23	A_2	A_2	A_2
D_0	D_0	D_0	19 22	A_1	A_1	A_1
$\overline{OE}$	$\overline{OE}$	$\overline{OE}$	20 21	A_0	A_0	A_0

Figure 2.14 : Brochage du boîtier DIL 40 pattes normalisé.

Le synoptique logique d'une 27C256 vous est présenté figure 2.15. Nous y voyons :

- Les deux pattes d'alimentation normales GND pour la masse et V_{CC} pour le +5 volts.

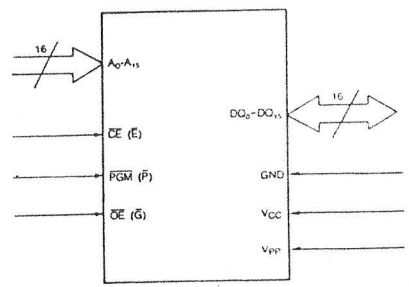


Figure 2.15 :
Synoptique logique d'une 27C256.

- La patte d'alimentation haute tension V_{PP} utilisée en phase de programmation.
- Les lignes d'adresses A_0 à A_{14}.
- Les lignes des données DQ_0 à DQ_7
- Une ligne de validation du boîtier CE barre active au niveau bas.
- Une ligne de validation des sorties OE barre, active elle aussi au niveau bas.

Celui d'une 27C1024 est, quant à lui, représenté figure 2.16. Un certain nombre de lignes sont du même type que celles de la 27C256 avec :

- Les mêmes lignes d'alimentation.
- Les adresses de A_0 à A_{15} (en raison de la taille plus importante de la mémoire).
- Les données DQ_0 à DQ_{15} puisque cette fois-ci la mémoire est organisée en mots de 16 bits.
- Les lignes CE barre et OE barre qui jouent le même rôle.
- Seule la ligne PGM, qui permet de placer la mémoire en mode programmation, est nouvelle par rapport au cas précédent. Nous allons voir que cela ne modifie que très peu l'utilisation de cette mémoire, comparativement à celle de la 27C256.

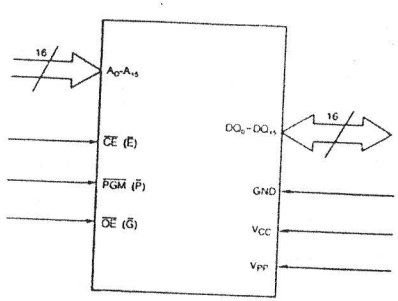


Figure 2.16 :
Synoptique logique d'une 27C1024.

LES DIFFÉRENTS MODES DE FONCTIONNEMENT

Toutes les mémoires UVPROM actuelles disposent de plusieurs modes de fonctionnement d'inégale importance. Les deux principaux sont évidemment le mode de repos, lorsque la mémoire n'est pas adressée dans un système par exemple, et le mode lecture puisque c'est tout de même cela la fonction première d'une UVPROM. Il existe néanmoins un certain nombre d'autres modes liés aux possibilités de programmation. Ceux-ci sont activés par mise aux niveaux adéquats des diverses pattes, autres que les adresses et les données, présentes sur les boîtiers. Dans la fiche technique de chaque mémoire un tableau permet de savoir comment y accéder.

Si nous reprenons notre exemple précédent, à savoir celui d'une 27C256 et celui d'une 27C1024, on se trouve en présence des tableaux présentés figures 2.17 et 2.18.

Comme vous pouvez le constater, ces deux tableaux diffèrent légèrement par la façon d'accéder aux divers modes de fonctionnement pour la simple raison que la 27C256 ne dispose pas de la patte PGM, alors que celle-ci est présente sur la 27C1024. Hormis cette diffé-

Figure 2.17 :
Table de sélection de mode d'une 27C256.

Mode \ Pins		$\overline{CE}$	$\overline{OE}$	A_0	A_9	V_{PP}	Outputs
Read		V_{IL}	V_{IL}	X	X	V_{CC}	D_{OUT}
Output Disable		V_{IL}	V_{IH}	X	X	V_{CC}	High Z
Standby (TTL)		V_{IH}	X	X	X	V_{CC}	High Z
Standby (CMOS)		V_{CC} ± 0.3 V	X	X	X	V_{CC}	High Z
Program		V_{IL}	V_{IH}	X	X	V_{PP}	D_{IN}
Program Verify		V_{IH}	V_{IL}	X	X	V_{PP}	D_{OUT}
Program Inhibit		V_{IH}	V_{IH}	X	X	V_{PP}	High Z
Auto Select (Notes 3 & 4)	Manufacturer Code	V_{IL}	V_{IL}	V_{IL}	V_H	V_{CC}	01 H
	Device Code	V_{IL}	V_{IL}	V_{IH}	V_H	V_{CC}	10 H

Notes: 1. X can be either V_{IL} or V_{IH}
2. V_H = 12.0 V ± 0.5 V
3. $A_1 - A_8 = A_{10} - A_{12} = V_{IL}$
4. A_{13} and A_{14} = X
5. See DC Programming characteristics for V_{PP} voltage during programming.

MODE \ PINS		$\overline{CE}$	$\overline{OE}$	$\overline{PGM}$	A_0	A_9	V_{PP}	OUTPUTS
Read		V_{IL}	V_{IL}	V_{IH}	X	X	V_{CC}	D_{OUT}
Output Disable		V_{IL}	V_{IH}	V_{IH}	X	X	V_{CC}	High Z
Standby (TTL)		V_{IH}	X	X	X	X	V_{CC}	High Z
Standby (CMOS)		V_{CC} ±0.3 V	X	X	X	X	V_{CC}	High Z
Program		V_{IL}	X	V_{IL}	X	X	V_{PP}	D_{IN}
Program Verify		V_{IL}	V_{IL}	V_{IH}	X	X	V_{PP}	D_{OUT}
Program Inhibit		V_{IH}	X	X	X	X	V_{PP}	High Z
Auto Select (Note 3)	Manufacturer Code	V_{IL}	V_{IL}	V_{IH}	V_{IL}	V_H	V_{CC}	01H
	Device Code	V_{IL}	V_{IL}	V_{IH}	V_{IH}	V_H	V_{CC}	8CH

Notes: 1. X can be either V_{IL} or V_{IH}
2. V_H = 12.0 V±0.5 V
3. A_1-A_8 = A_{10} – A_{15} = V_{IL}
4. See DC Programming Characteristics for V_{PP} voltage during programming.

Figure 2.18 :
Table de sélection de mode d'une 27C1024.

rence, les modes proposés sont identiques et correspondent aux fonctionnalités suivantes :

- Mode lecture (*read*) : la mémoire est validée et la donnée contenue à l'adresse présente sur A_0 à A_X est fournie sur les lignes D_0 à D_Y placées en sorties.
- Mode invalidation des sorties (*output disable*) : la mémoire est validée et l'adressage est pris en compte, mais les données ne sont pas appliquées aux lignes de sorties qui restent en haute impédance. Dès la descente de OE, la donnée apparaît sur les sorties.
- Mode attente TTL (*standby TTL*) : la mémoire est invalidée. L'adresse appliquée sur A_0 à A_X est ignorée. Les sorties sont en haute impédance et la consommation est réduite à 1 mA.
- Mode attente CMOS (*standby CMOS*) : ce mode est analogue au précédent mais la consommation de la mémoire est réduite à 100 µA.
- Mode programmation (*program*) : la mémoire est validée en programmation. Sous réserve de respecter les chronogrammes corrects (présentés ci-après) la donnée présentée sur D_0 à D_Y est écrite à l'adresse sélectionnée par A_0 à A_X.
- Mode vérification (*verify*) : la mémoire passe en lecture, bien que la haute tension de programmation lui soit toujours appliquée. Cela

permet de vérifier très rapidement la programmation correcte ou incorrecte de la donnée. Ce mode ne doit être utilisé qu'en phase de vérification lors de la programmation, et en aucun cas en lecture normale (on ne voit d'ailleurs pas très bien quel serait l'intérêt d'une telle démarche).

- Mode inhibition de programmation (*program inhibit*) : ce mode n'est utilisé que sur certains programmateurs particuliers où diverses mémoires sont programmées simultanément mais avec des données pouvant être différentes. Il permet la sélection facile d'une mémoire parmi N qui sera alors la seule à être programmée.
- Mode identification automatique (*auto select*) : ce mode qui, à l'heure où ces lignes sont écrites, peut ne pas exister encore sur toutes les mémoires surtout de petite taille, permet de lire une zone particulière de la mémoire dans laquelle est codée de manière unique le nom du fabricant de la mémoire et le type de mémoire. Dans ces conditions, les lignes de données délivrent un code lorsque A_0 est à 0 qui est le code du fabricant de la mémoire et un autre code lorsque A_0 est à 1 qui est le code du type de mémoire.

Ce mode est extrêmement intéressant pour les programmateurs intelligents. En effet, si ces derniers savent l'exploiter ils peuvent ainsi se configurer exactement comme il faut compte tenu de la mémoire à programmer et l'utilisateur n'a plus à se soucier de la marque et de la référence exacte de la mémoire.

Ce mode porte divers noms selon le fabricant de la mémoire : *intelligent identifier* chez Intel, *auto select mode* chez AMD, *silicon signature* chez SEEQ, Malgré ces divergences d'appellations, il fonctionne fort heureusement de la même façon chez tout le monde, ce qui est le moins que l'on puisse demander si on veut lui conserver son intérêt.

Ceci étant précisé, l'utilisation en lecture d'une UVPROM ne présente aucune spécificité. Le chronogramme de la figure 2.19 est en effet des plus classiques.

Les adresses sont appliquées sur A_0 à A_X et, dès leur stabilisation, la mémoire est validée par mise au niveau bas de la ligne CE barre. Aussitôt ou à n'importe quel instant ultérieur, la ligne OE barre peut être mise au niveau bas pour valider les sorties sur lesquelles la don-

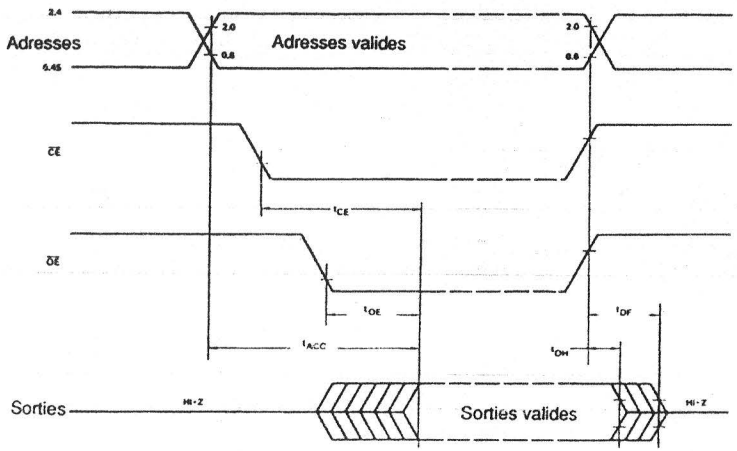


Figure 2.19 :
Chronogramme de lecture d'une 27C256.

née sélectionnée est disponible après respect du temps d'accès minimum t_{acc}. À titre d'exemple, la figure 2.20 indique les divers temps relatifs à cette figure dans le cas d'une 27C256 d'AMD.

Parameter Symbols		Parameter Description	Test Conditions		Am27C256										Unit
JEDEC	Standard				-55	-70, -75	-90, -95	-100, -105	-120, -125	-150, -155	-170, -175	-200, -205	-250, -255	-300, -305	
t_{AVQV}	t_{ACC}	Address to Output Delay	$\overline{CE} = \overline{OE} = V_{IL}$	Min.											ns
				Max.	55	70	90	100	120	150	170	200	250	300	
t_{ELQV}	t_{CE}	Chip Enable to Output Delay	$\overline{OE} = V_{IL}$	Min.											ns
				Max.	55	70	90	100	120	150	170	200	250	300	
t_{GLQV}	t_{OE}	Output Enable to Output Delay	$\overline{CE} = V_{IL}$	Min.											ns
				Max.	35	40	40	40	50	65	70	75	100	120	
t_{EHQZ}, t_{GHQZ}	t_{DF}	Output Enable HIGH to Output Float (Note 2)		Min											ns
				Max.	25	25	25	30	35	50	50	55	60	75	
t_{AXQX}	t_{OH}	Output Hold from Addresses, $\overline{CE}$, or $\overline{OE}$, whichever occurred first		Min.	0	0	0	0	0	0	0	0	0	0	ns
				Max											

Figure 2.20 :
Les temps de la figure 2.19 pour les diverses versions de 27C256 disponibles.

PROGRAMMATION DES UVPROM

Il était courant, lors de l'utilisation des UVPROM de petite taille qu'étaient les 2716 et 2732, de réaliser son propre programmateur lorsque l'on n'avait pas sous la main l'appareil adéquat. Les chronogrammes à respecter étaient fort simples et même si le temps de programmation était loin d'être négligeable (de l'ordre de 100 ms par octet tout compris) il restait dans les limites du raisonnable. Une 2732 avec ses 4 Kmots de 8 bits se programmait alors en 400 secondes environ soit 6 à 7 minutes.

L'arrivée sur le marché de mémoires de tailles plus importantes, conjointement à des modifications des architectures des cellules mémoires ont conduit les fabricants à proposer d'autres algorithmes de programmation beaucoup plus rapides. En effet, l'algorithme standard avec ses 100 ms par octet est totalement inutilisable en production avec des mémoires telles que les 27512 par exemple, car il conduit à un temps de programmation de 6400 secondes soit près de deux heures !

Des algorithmes rapides ont donc été proposés. Sauf problème particulier, ils permettent de réduire les temps de programmation des plus grosses mémoires à quelques secondes, voire au maximum à une minute environ. En contrepartie, ils sont plus complexes à mettre en œuvre et ne se rencontrent donc bien souvent que sur les programmateurs commerciaux traditionnels.

Ces algorithmes rapides ont, à leurs débuts, posé beaucoup de problèmes aux constructeurs de programmateurs car ils différaient d'un fabricant à l'autre et parfois d'une mémoire à l'autre chez un même fabricant. Certaines tensions d'alimentation variaient (6 volts chez l'un et 6,25 volts chez l'autre ; certains temps étaient notablement différents (10 µs chez l'un et 1 ms chez l'autre), ce qui compliquait bien évidemment d'autant la conception des programmateurs qui se voulaient universels.

Cette situation n'a heureusement plus cours aujourd'hui et, même si le nom des algorithmes utilisés diffère encore d'un fabricant à l'autre, les temps, les tensions et l'enchaînement des opérations sont identiques.

Ceci étant précisé, voici à titre d'exemple la procédure de programmation rapide générale d'une UVPROM type 27C256.

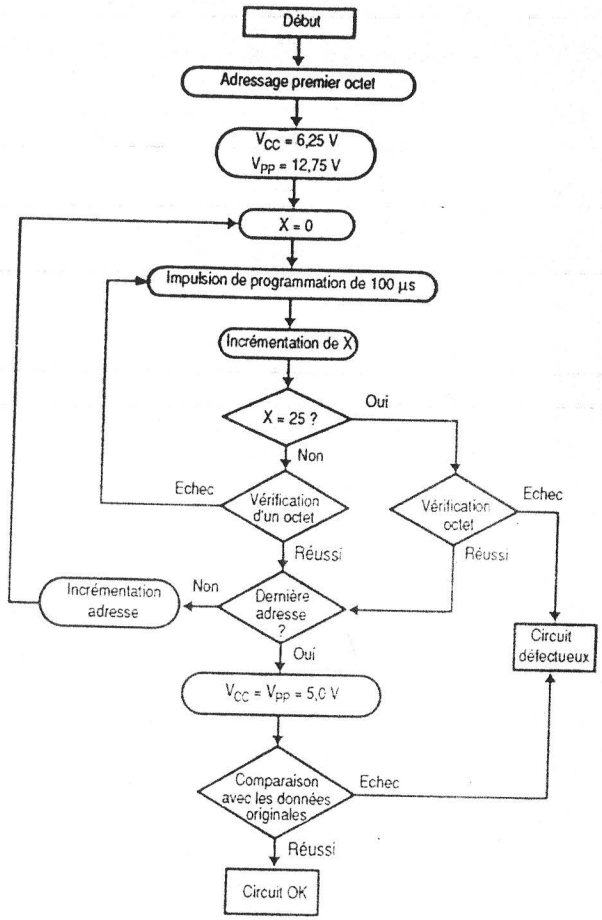


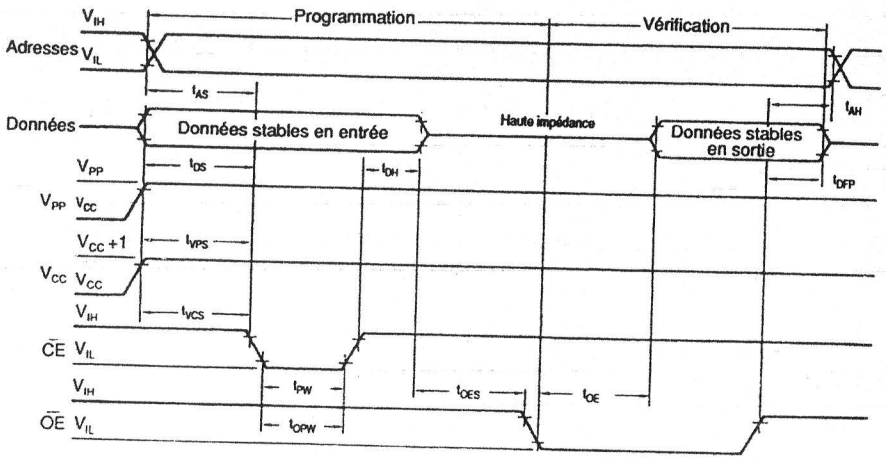
Figure 2.21 :
Algorithme de programmation rapide d'une 27C256 Intel.

Tout d'abord, il faut savoir que les UVPROM vierges ont tous leurs bits positionnés à 1. La procédure de programmation consiste donc juste à programmer les 0. Ceci n'influe pas sur la démarche générale, mais permet de savoir que l'on peut se dispenser d'effacer une mémoire pour la reprogrammer ensuite si l'on a juste quelques bits à faire passer de 1 à 0.

La mémoire doit en premier lieu être placée en mode programmation. Pour ce faire, il faut se reporter au tableau de sélection des modes de fonctionnement pour savoir comment y parvenir. Celui de la 27C256 choisie en exemple et présenté figure 2.17 montre que cet état est obtenu lorsque la haute tension de programmation de 12,75 volts est appliquée à la patte V_{PP}, et lorsque la tension d'alimentation est portée de 5,0 à 6,25 volts.

Comme le montre l'organigramme de la figure 2.21, l'adresse de la première cellule mémoire à programmer est sélectionnée et la donnée à programmer est appliquée sur D_0 à D_7. Une impulsion de programmation d'une durée de 100 µs est alors appliquée. Cette impulsion est en fait un passage au niveau bas de la ligne CE barre comme le montre le chronogramme de la figure 2.22. Un compteur est alors incrémenté et la mémoire est relue. Si la valeur lue est conforme à celle programmée on passe à l'adresse suivante. Dans le cas contraire, une nouvelle impulsion de 100 µs est à nouveau appliquée, le compteur est incrémenté et la mémoire est relue. Ce processus peut se répéter jusqu'à 25 fois au maximum. Si passé ce nombre, la mémoire n'est toujours pas correctement programmée, elle doit être considérée comme défectueuse, au moins pour ce qui est de l'adresse sur laquelle on se trouve (ce qui en général revient au même sauf si la mémoire n'est que très partiellement utilisée).

Si tout s'est déroulé comme il faut, on passe à l'adresse suivante. Bien que l'organigramme de la figure 2.21 indique une programmation suivant le sens des adresses croissantes, rien ne vous oblige à programmer toute la mémoire d'un coup. Si seules quelques adresses sont nécessaires, c'est tout à fait possible, et ce, dans n'importe quel ordre et même à plusieurs mois ou années d'intervalle. Après cela, une ultime lecture est réalisée pour s'assurer de la programmation correcte de la mémoire.



Parameter Symbols		Parameter Description	Min	Max	Unit
JEDEC	Standard				
tAVEL	tAS	Address Setup Time	2		µs
tDZGL	tOES	$\overline{OE}$ Setup Time	2		µs
tDVEL	tDS	Data Setup Time	2		µs
tGHAX	tAH	Address Hold Time	0		µs
tEHDX	tDH	Data Hold Time	2		µs
tGHQZ	tDFP	Output Enable to Output Float Delay	0	130	ns
tVPS	tVPS	VPP Setup Time	2		µs
tELEH	tPW	$\overline{PGM}$ Program Pulse Width	95	105	µs
tVCS	tVCS	VCC Setup Time	2		µs
tGLQV	tOE	Data Valid from $\overline{OE}$		150	ns

Figure 2.22 : Chronogramme de programmation rapide d'une 27C256.

On le voit, sans être particulièrement complexe, ce processus ne peut valablement s'exécuter de façon fiable et sûre que sous le contrôle d'un microprocesseur ou microcontrôleur et ce d'autant plus que de nombreux paramètres doivent parfois être modifiés, comme nous l'avons vu, en fonction des mémoires utilisées et de leurs fabricants.

LES ONE TIME PROM OU OTP

Les UVPROM ont de nombreuses qualités qui expliquent leur succès considérable, mais souffrent en contrepartie d'un défaut majeur : le coût... de leur boîtier. Cela peut en effet sembler curieux, mais, dans bien des UVPROM, c'est le boîtier qui coûte cher en raison de la fenêtre en quartz dont il doit être muni pour permettre l'effacement de la mémoire.

Pour des productions unitaires ou en très petite série, ce coût est sans grande importance, mais prend des proportions dramatiques lors de la production de série. Pour remédier à cela, les fabricants ont alors eu l'idée de placer des UVPROM dans des boîtiers sans fenêtre qui étaient de ce fait bien moins coûteux à fabriquer. Les mémoires ont évidemment conservé toutes les possibilités des UVPROM sauf une : l'effacement. Elles sont devenues des mémoires programmables une fois ou OTP en bon anglais !

Pour étrange qu'elle puisse être, cette approche présente de nombreux avantages :

- Baisse du prix de revient par rapport à une UVPROM classique.
- Utilisation du même programmateur qu'une UVPROM classique.
- Caractéristiques électriques identiques à celles d'une UVPROM classique.

Le fait d'être non effaçable n'est aucunement gênant en production de série puisque, dans ce cas, les données mises dans la PROM sont en général figées.

Nous ne vous proposerons aucun tableau de référence ou de sélection de ces mémoires, car chez la majorité des fabricants, toutes les mémoires UVPROM sont disponibles en version OTP. Il suffit de changer un suffixe de la référence de l'UVPROM pour trouver celle de la version OTP. Les brochages et modes de fonctionnement sont évidemment identiques comme nous venons de l'expliquer.

LES PROM EFFAÇABLES ÉLECTRIQUEMENT OU EEPROM

Théoriquement nous devrions vous présenter sous ce titre les EEPROM appelées aussi E2PROM ainsi que les EPROM FLASH qui sont également effaçables électriquement.

Bien qu'à priori semblables, de telles mémoires sont technologiquement différentes et surtout ne s'utilisent pas de la même façon et n'ont pas les mêmes possibilités. Nous avons donc estimé plus judicieux de vous les présenter indépendamment les unes des autres.

Les PROM effaçables électriquement ou EEPROM sont arrivées sur le marché bien après que les UVPROM aient commencé à acquérir leurs lettres de noblesse. Le pionnier en la matière a été l'américain Xicor qui reste encore le fabricant proposant les plus nombreuses références en ce domaine.

Ces mémoires se présentent, au plan de l'organisation et du brochage, comme des UVPROM dont elles respectent d'ailleurs l'organisation en mots de 8 bits, tout au moins pour celles à utilisation micro-informatique.

Une autre famille d'EEPROM connaît un grand succès actuellement, celle des EEPROM à accès série. Ces mémoires, généralement de taille relativement réduite (quelques centaines d'octets à quelques Ko) servent à mémoriser de façon semi-permanente des informations entrées par l'utilisateur dans les appareils les plus divers. Ce sont elles qui mémorisent les chaînes de votre récepteur TV ou de votre magnétoscope par exemple.

Qu'elles soient de type série ou parallèle, ces mémoires utilisent la même technologie, aussi ne présenterons-nous que les mémoires organisées en mots de 8 bits.

Ces mémoires se programment presque comme les UVPROM, à quelques détails près, mais à ce stade de l'ouvrage vous devez commencer à en avoir l'habitude. Elles présentent en outre la possibilité de pouvoir être effacées électriquement dans leur ensemble, ou octet par octet si nécessaire. C'est ce dernier point qui les différencie des mémoires FLASH qui ne sont, elles, effaçables que globalement.

Enfin et surtout, elles ne font appel, tant pour la programmation que pour l'effacement, à aucune haute tension particulière et travaillent donc avec une seule et unique source de 5 volts. Ici encore, cela les

différencie des mémoires FLASH qui ont besoin d'une *haute tension* de programmation, encore que les FLASH 5 volts commencent à arriver sur le marché.

À première vue, ces mémoires sont donc séduisantes puisqu'il n'y a plus besoin de source d'ultra-violets pour les effacer et de l'inévitable perte de temps que cela génère, ni de source de haute tension. De ce fait, elles peuvent être programmées et effacées en *circuit*, pour peu que la logique nécessaire ait été prévue, ce qui ouvre des horizons nouveaux aux concepteurs.

Ces mémoires, à priori si parfaites qu'on serait en droit de se demander pourquoi existent encore les UVPROM, souffrent malheureusement d'un défaut notable : la cellule mémoire élémentaire est nettement plus complexe que dans une UVPROM, ce qui augmente notablement leur coût de fabrication et limite leur capacité, comparativement aux UVPROM bien sûr. En outre, l'émergence de la technologie FLASH dont nous avons déjà dit quelques mots et que nous vous présenterons en détail dans un instant leur a fait perdre une partie de leur intérêt.

Ces mémoires EEPROM ne se retrouvent donc plus qu'en très petit nombre dans les catalogues des grands fabricants de mémoires ; ainsi, AMD ne propose-t-il plus au moment où ces lignes sont écrites aucune référence d'EEPROM. Il est donc nécessaire de s'adresser à des fabricants spécialisés tels que Xicor pour trouver un choix raisonnable, ou de faire appel à la technologie FLASH où le choix et la capacité mémoire ne cessent de s'accroître.

CARACTÉRISTIQUES PRINCIPALES

Les durées de rétention d'information sont analogues à celles des UVPROM, à savoir au minimum 10 ans à 70 °C tandis que le nombre maximum de cycles d'effacement ou de programmation (ce qui est la même chose) est de 10000 comme pour les UVPROM.

Pour ce qui est de l'utilisation et du brochage, les EEPROM sont identiques aux UVPROM de même taille si ce n'est la présence d'une ligne appelée WE pour write enable (autorisation d'écriture) qui prend tout naturellement la place de la ligne PGM barre des mémoires

UVPROM de même taille. En effet, une autre particularité de ces mémoires est de savoir générer en interne leur propre chronogramme de programmation.

À titre d'exemple, la figure 2.23 présente le brochage d'une 2864 qui est une EEPROM identique à la référence 27C64 en UVPROM. Sur cette même figure vous est d'ailleurs proposé également le tableau de sélection de mode de fonctionnement. On y retrouve un certain nombre de modes rigoureusement identiques à ceux vus pour les UVPROM et nous n'y reviendrons donc pas.

En lecture, ces mémoires s'utilisent exactement comme des UVPROM. Vous pouvez donc vous reporter intégralement à ce que nous avons écrit précédemment. Seule la programmation diffère comme nous allons le voir maintenant.

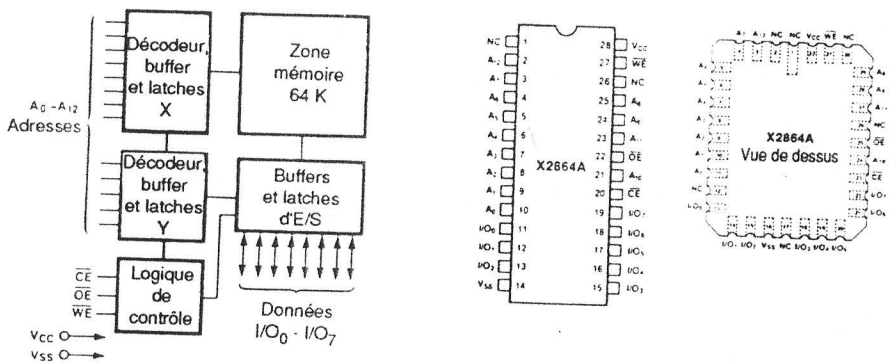


Figure 2.23 : *Brochage, synoptique et tableau de sélection de mode d'une EEPROM X2864 de Xicor.*

Sélection de mode

$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	Mode	I/O	Etat
L	L	H	Lecture	D_{OUT}	Actif
L	H	L	Ecriture	D_{IN}	Actif
H	X	X	Repos et interdiction d'écriture	Haute impédance	Repos
X	L	X	Interdiction d'écriture	—	—
X	X	H	Interdiction d'écriture	—	—

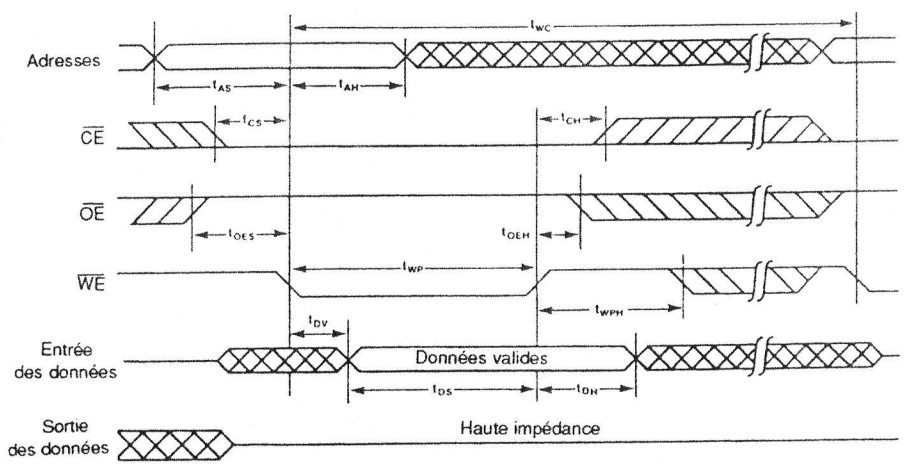
PROGRAMMATION DES EEPROM

La programmation d'une EEPROM, qu'il est d'ailleurs presque normal d'appeler une écriture dans la mémoire, se passe comme l'écriture dans une mémoire vive à ceci près, qu'au lieu d'être quasi-immédiate, elle demande environ 10 ms par octet (ou par groupe de 32 octets comme nous le verrons après).

Contrairement à une UVPROM qu'il faut mettre en mode programmation par application d'une haute tension et combinaison de niveaux logiques sur certaines pattes, une EEPROM passe immédiatement du mode lecture au mode programmation par simple changement d'état de la ligne d'écriture WE. Cela ressort d'ailleurs clairement de l'examen du tableau de la figure 2.23.

Le chronogramme d'écriture d'un octet est présenté pour mémoire figure 2.24 et est identique à un cycle d'écriture dans une RAM. En effet on positionne les adresses, on valide la mémoire par action sur CE barre, on invalide les sorties (puisque l'on va écrire dans la mémoire) par action sur OE barre puis on fait descendre WE barre et on applique les données à écrire sur les sorties D_0 à D_7. Ces données sont mémorisées lors du front montant suivant de WE barre et le

Figure 2.24 :
Chronogramme d'écriture ou de programmation d'une EEPROM.



cycle de programmation interne commence alors. Il faut évidemment attendre qu'il soit terminé avant de programmer un autre octet. Pour ce faire, soit un dispositif externe génère une boucle d'attente de durée suffisamment longue pour dépasser le temps maximum de programmation indiqué par le fabricant, soit le mode *data polling* est utilisé. Ce mode permet en effet de surveiller le déroulement du cycle de programmation interne et de savoir exactement quand il est terminé. Il fonctionne de la façon suivante.

Ce mode est déclenché dès la fin d'un cycle de programmation par mise au niveau haut de la patte WE barre et par mise au niveau bas de la patte OE barre. Les données qui sont alors lues sont l'inverse logique (1 à la place de 0 et *vice versa*) des données qui ont été programmées tant que le cycle interne n'est pas terminé. Dès que celui ci est fini, les données redeviennent vraies. La figure 2.25 précise cela avec un chronogramme aussi bien qu'un long discours.

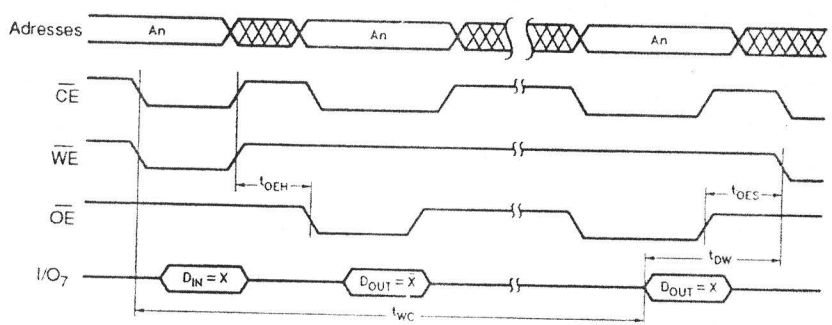


Figure 2.25 :
Principe de l'indication de fin d'écriture par inversion des données dans une EEPROM.

Comme la programmation d'un octet prend environ 10 ms, un mode dit de programmation par page existe. Il permet de programmer d'un seul coup 32 octets dans ce seul et même temps de 10 ms. Comme le montre le chronogramme de la figure 2.26, les adresses et données relatives à ces 32 octets sont chargées dans la mémoire lors de chaque front montant de WE barre. Dès que le nombre maximum de 32 est atteint, la mémoire passe en mode programmation et réalise alors la programmation de ces 32 données aux 32 adresses spécifiées. Ces adresses peuvent être quelconques et n'ont pas besoin d'être contigües

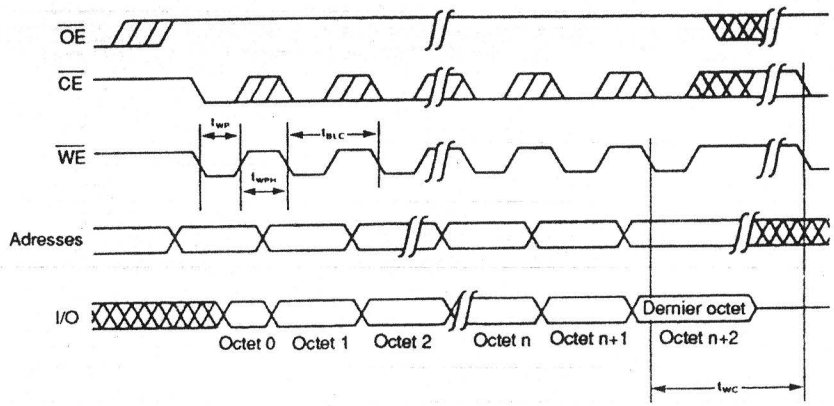


Figure 2.26 :
Chronogramme de programmation d'une EEPROM en mode page.

ou placées dans un ordre pré-défini. C'est donc là une procédure très souple qui permet de gagner beaucoup de temps lorsque l'on désire programmer un nombre important d'informations dans la mémoire.

Précisons, pour en terminer avec ces EEPROM, que si vous souhaitez les programmer sur un programmateur, ce qui reste tout à fait possible pour peu que celui-ci les accepte, elles supportent la procédure d'identification automatique que nous avons vue pour les UVPROM. Cette procédure est évidemment déclenchée de la même façon, la patte WE barre jouant alors le rôle de la patte PGM.

LES MÉMOIRES FLASH OU FLASH EPROM

Comme leur sigle le laisse supposer et compte tenu de ce que nous vous avons expliqué précédemment, les FLASH EPROM sont des mémoires programmables et effaçables électriquement. Qu'est ce qui les distingue des EEPROM, alors nous direz-vous ? Tout simplement le fait qu'elles soient plus rapides (en termes de programmation et d'effacement), d'où l'appellation FLASH, mais aussi le fait qu'elles ne permettent que l'effacement total de la mémoire. En outre, comme les UVPROM, elles utilisent encore pour la majorité d'entre elles une haute tension de programmation de 12 volts (12,75 volts pour les premières versions en voie de disparition) même si des versions monotension 5 volts commencent à voir le jour.

Malgré cet apparent retour en arrière, les FLASH EPROM présentent de nombreux avantages qui justifient leur succès foudroyant.

En effet, ces mémoires, du fait de leur technologie beaucoup plus simple que celle des EEPROM, coûtent nettement moins cher et, surtout, existent avec des capacités supérieures, proches de ce que l'on sait faire avec des UVPROM.

En outre, les temps de programmation et d'effacement sont remarquables : 100 µs de temps de programmation par octet et 1 seconde pour l'effacement total d'une 28F256 chez Intel par exemple (les autres fabricants arrivent à des temps similaires).

CARACTÉRISTIQUES PRINCIPALES

Le tableau de la figure 2.27 donne la liste des mémoires FLASH EPROM proposées par Intel à la date de rédaction de cet ouvrage. On constate que les capacités offertes atteignent presque celles des UVPROM et sont nettement plus importantes que celles permises par les EEPROM. C'est parfaitement logique lorsque l'on examine la technologie de ces mémoires qui est très proche de celle des UVPROM. En effet, alors que les EEPROM ont besoin de plusieurs transistors par cellule mémoire élémentaire, les FLASH EPROM n'utilisent, comme les UVPROM, qu'un transistor MOS par bit, même s'il est un tout petit peu plus complexe, il est vrai.

Remarquez que ces mémoires existent tout à la fois sous forme de composants classiques en boîtiers DIL ou LCC, mais aussi sous forme de barrettes SIM ou de cartes PCMCIA. L'industrie des compatibles PC est passée par là !

Notez que le boîtier retenu, en DIL ou LCC, est le modèle 32 pattes que nous avons déjà vu pour les plus grosses UVPROM. Pour les plus petites FLASH EPROM, les pattes surnuméraires sont non connectées comme vous pouvez le constater à l'examen de la figure 2.28. Les brochages sont évidemment strictement compatibles avec ceux des UVPROM au changement de dénomination près d'une ligne : la patte de programmation PGM devient en effet WE pour *Write Enable* (autorisation d'écriture).

Reference	Capacite	Boitier	Nb de pattes	5,0 V_{CC}	3,3 V_{CC}
FlashFile™ MEMORY COMPONENTS - Symmetrically blocked architecture					
*28F032SA	32Mb, 4M x 8/ 2M x 16	DD	56	70, 100	120, 150
*28F016SA	16Mb, 2M x 8/ 1M x 16	E	56	70, 100	120, 150
28F008SA	8Mb, 1M x 8	E,F	40	85, 120	200
		PA	44	85, 120	200
BOOT BLOCK FLASH MEMORY COMPONENTS - Asymmetrically blocked architecture					
28F400BX	4Mb, 512K x 8/ 256K x 16	PA	44	60, 80	150
		E	56	60, 80	150
28F004BX	4Mb, 512K x 8	E	40	60, 80	150
28F200BX	2Mb, 256K x 8/ 128K x 16	PA	44	60, 80	150
		E	56	60, 80	150
28F002BX	2Mb, 256K x 8	E	40	60, 80	150
28F001BX	1Mb, 128K x 8	P,N,E	32	TBD, 120, 150	
FLASH MEMORY COMPONENTS - Bulk-Array					
28F020	2Mb, 256K x 8	P,N,E,F	32	70, 90, 150, 200	
28F010	1Mb, 128K x 8	E,F	32	65, 90, 120, 150	
		P,N	32	65, 90, 120, 150	
28F512	512K, 64K x 8	P,N	32	120, 150, 200	
28F256A	256K, 32K x 8	P,N	32	120, 150, 200	
SERIES 2+ FLASH MEMORY CARDS - PCMCIA Standard Interface					
*iMC040FLSP	40MB, x8/ x16	MC	68	150	200
*iMC020FLSP	20MB, x8/ x16	MC	68	150	200
*iMC004FLSP	4MB, x8/ x16	MC	68	150	200
SERIES 2 FLASH MEMORY CARDS - PCMCIA Standard Interface					
iMC020FLSA	20MB, x8/ x16	MC	68	200	
iMC010FLSA	10MB, x8/ x16	MC	68	200	
iMC004FLSA	4MB, x8/ x16	MC	68	200	
iMC002FLSA	2MB, x8/ x16	MC	68	200	
SERIES 1 FLASH MEMORY CARDS - PCMCIA Standard Interface					
iMC004FLKA	4MB, x8/ x16	MC	68	200	
iMC002FLKA	2MB, x8/ x16	MC	68	200	
iMC001FLKA	1MB, x8/ x16	MC	68	200	
FLASH DRIVE - PCMCIA - ATA Interface					
*iFD010P2SA	10MB	Type 2	68	<1ms	
*iFD005P2SA	5MB	Type 2	68	<1ms	

Figure 2.27 :
Les mémoires FLASH commercialisées par Intel à la date de rédaction de cet ouvrage.

28F020	28F010	28F512	28F256		28F256	28F512	28F010	28F020
V_{PP}	V_{PP}	V_{PP}	V_{PP}	1 / 32	V_{CC}	V_{CC}	V_{CC}	V_{CC}
A_{16}	A_{16}	–	–	2 / 31	$\overline{WE}$	$\overline{WE}$	$\overline{WE}$	$\overline{WE}$
A_{15}	A_{15}	A_{15}	–	3 / 30	–	–	–	A_{17}
A_{12}	A_{12}	A_{12}	A_{12}	4 / 29	A_{14}	A_{14}	A_{14}	A_{14}
A_7	A_7	A_7	A_7	5 / 28	A_{13}	A_{13}	A_{13}	A_{13}
A_6	A_6	A_6	A_6	6 / 27	A_8	A_8	A_8	A_8
A_5	A_5	A_5	A_5	7 / 26	A_9	A_9	A_9	A_9
A_4	A_4	A_4	A_4	8 / 25	A_{11}	A_{11}	A_{11}	A_{11}
A_3	A_3	A_3	A_3	9 / 24	$\overline{OE}$	$\overline{OE}$	$\overline{OE}$	$\overline{OE}$
A_2	A_2	A_2	A_2	10 / 23	A_{10}	A_{10}	A_{10}	A_{10}
A_1	A_1	A_1	A_1	11 / 22	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$
A_0	A_0	A_0	A_0	12 / 21	D_7	D_7	D_7	D_7
D_0	D_0	D_0	D_0	13 / 20	D_6	D_6	D_6	D_6
D_1	D_1	D_1	D_1	14 / 19	D_5	D_5	D_5	D_5
D_2	D_2	D_2	D_2	15 / 18	D_4	D_4	D_4	D_4
Masse	Masse	Masse	Masse	16 / 17	D_3	D_3	D_3	D_3

Figure 2.28 :
Brochage normalisé du boîtier DIL 32 pattes des mémoires FLASH.

Comme pour les UVPROM, la durée de rétention garantie de l'information est au moins de 10 ans à 70 °C et elles supportent au minimum 10000 cycles de programmation/effacement.

Même si la structure interne de telles mémoires est relativement complexe comme le montre à titre d'exemple celle de la 28F256 présentée figure 2.29, l'utilisation en lecture de ces mémoires est identique à celle des UVPROM.

Comme pour les UVPROM et EEPROM, les diverses combinaisons de niveaux applicables à certaines pattes autorisent la sélection de divers modes de fonctionnement qui sont présentés, toujours dans le cas de la 28F256, figure 2.30. Tous les modes proposés vous sont désormais connus, sauf le mode écriture ou programmation dont nous allons voir les particularités dans le cas d'une telle mémoire. Notez tout de même la présence du processus d'identification automatique qui est activé comme sur toutes les autres mémoires qui le supportent.

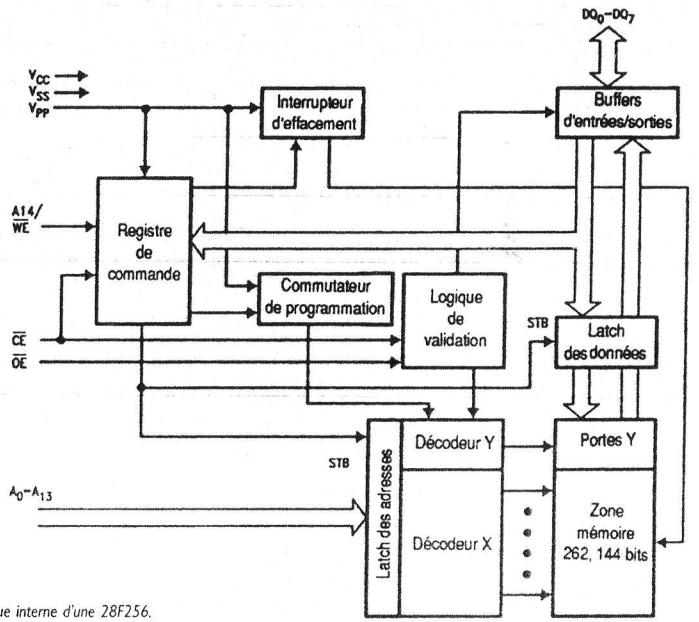


Figure 2.29 : Synoptique interne d'une 28F256.

	Pins / Operation	V_{PP} (1)	A_0	A_9	$\overline{CE}$	$\overline{OE}$	A14/ $\overline{WE}$	DQ_0–DQ_7
READ-ONLY	Read	V_{PPL}	A_0	A_9	V_{IL}	V_{IL}	A_{14}	Data Out
	Output Disable	V_{PPL}	X	X	V_{IL}	V_{IH}	V_{IH}	Tri-state
	Standby	V_{PPL}	X	X	V_{IH}	X	X	Tri-state
	intelligent ID™ Manufacturer (2)	V_{PPL}	V_{IL}	V_{ID} (3)	V_{IL}	V_{IL}	V_{IL}	Data = 89H
	intelligent ID™ Device (2)	V_{PPL}	V_{IH}	V_{ID} (3)	V_{IL}	V_{IL}	V_{IL}	Data = 91H
READ/ WRITE	Read	V_{PPH}	A_0	A_9	V_{IL}	V_{IL}	V_{IH}	Data Out (4)
	Output Disable	V_{PPH}	X	X	V_{IL}	V_{IH}	V_{IH}	Tri-state
	Standby (5)	V_{PPH}	X	X	V_{IH}	X	X	Tri-state
	Write	V_{PPH}	A_0	A_9	V_{IL}	V_{IH}	V_{IL}	Data In (6)

Figure 2.30 :
Tableau de sélection de mode d'une 28F256.

PROGRAMMATION DES FLASH EPROM

Compte tenu des diverses possibilités offertes, et surtout du fait que les EPROM FLASH sont prévues pour être programmées et effacées en circuit, c'est-à-dire sans les enlever de l'application où elles sont utilisées, le mode programmation diffère un peu de celui vu précédemment pour les UVPROM et EEPROM. En effet, lorsque la mémoire est placée en mode programmation ou, plus exactement, lorsque la tension appliquée sur la patte V_{PP} est égale à V_{PPH} comme nous l'avons vu figure 2.30, on a accès à un registre interne dans lequel on peut écrire divers codes qui activent alors les différentes possibilités d'effacement et de programmation.

Ce registre est évidemment indépendant de l'espace mémoire adressable avec lequel il n'interfère absolument pas. Son mode d'accès et les données à y écrire se prêtent, par contre, remarquablement bien à l'utilisation d'une telle mémoire avec n'importe quel type de microprocesseur ou de microcontrôleur.

La figure 2.31 donne, à titre d'exemple, les différentes possibilités offertes par un 28F256 d'Intel en fonction du code écrit dans ce registre appelé registre de commande. On remarque tout de suite qu'il est possible d'ordonner la lecture de la mémoire ce qui signifie qu'il n'est pas utile de commuter la haute tension V_{PPH} de sur la patte

Command	Bus Cycles Req'd	First Bus Cycle			Second Bus Cycle		
		Operation(1)	Address(2)	Data(3)	Operation(1)	Address(2)	Data(3)
Read Memory	1	Write	X	00H			
Read int$_e$ligent ID Codes	3	Write	X	90H	Read	(4)	(4)
Set-Up Erase/Erase(6)	2	Write	X	20H	Write	X	20H
Erase Verify(6)	2	Write	EA	A0H	Read	X	EVD
Set-Up Program/Program(5)	2	Write	X	40H	Write	PA	PD
Program Verify(5)	2	Write	X	C0H	Read	X	PVD
Reset(7)	2	Write	X	FFH	Write	X	FFH

Figure 2.31 :
Utilisation du registre de commande d'une 28F256.

V_{PP}. En effet, la mémoire est en mode lecture lorsque la patte V_{PP} est au niveau V_{PPL} comme nous l'avons vu figure 2.30, mais elle est aussi en lecture lorsque la patte V_{PP} est au niveau V_{PPH} et que l'on écrit 00 dans le registre de commande. Ceci a été réalisé afin de faciliter l'utilisation de ces mémoires pour lesquelles il devient inutile de prévoir une commutation de la haute tension ; tous les modes de fonctionnement sont en effet accessibles, haute tension présente, par simple action sur le registre de commande.

Pour accéder à ce registre de commande, et puisqu'il n'est pas placé dans l'espace mémoire adressable comme nous venons de le dire, il faut utiliser le processus suivant, représenté sous forme de chronogramme figure 2.32.

La mémoire est mise en mode programmation par application de la haute tension V_{PPH} sur la patte V_{PP}. La donnée présente sur D_0 à D_7 est alors écrite dans le registre lorsque WE barre passe au niveau bas alors que CE barre est au niveau bas et OE barre au niveau haut bien sûr. Par défaut, et tant qu'il n'a pas été écrit dans le registre, celui-ci contient 00, suite à l'application de V_{PPH} sur V_{PP}. Ce code a pour effet de placer la mémoire en mode lecture comme nous l'avons vu figure 2.31, ce qui est une sécurité.

Ceci étant précisé, la lecture du tableau de la figure 2.31 ne pose pas de problème particulier, puisqu'on y retrouve les modes classiques : programmation, vérification de programmation, effacement et vérification d'effacement ainsi que la lecture si nécessaire de l'identificateur de mémoire comme sur les UVPROM et EEPROM.

Un algorithme *intelligent* ou rapide est évidemment utilisé pour la programmation, mais aussi pour l'effacement. Ceux-ci vous sont présentés sous forme d'organigrammes figures 2.33 et 2.34 dans le cas des mémoires FLASH Intel. Si ces mémoires sont programmées et effacées sur un programmateur classique, celui-ci doit bien évidemment respecter ces organigrammes. Si, par contre, ces mémoires sont programmées en *circuit*, c'est au microprocesseur ou au microcontrôleur qui pilote ces mémoires de générer ces diverses séquences grâce à un programme approprié. Cela ne présente aucune difficulté et ne demande que quelques dizaines d'instructions écrites en langage machine.

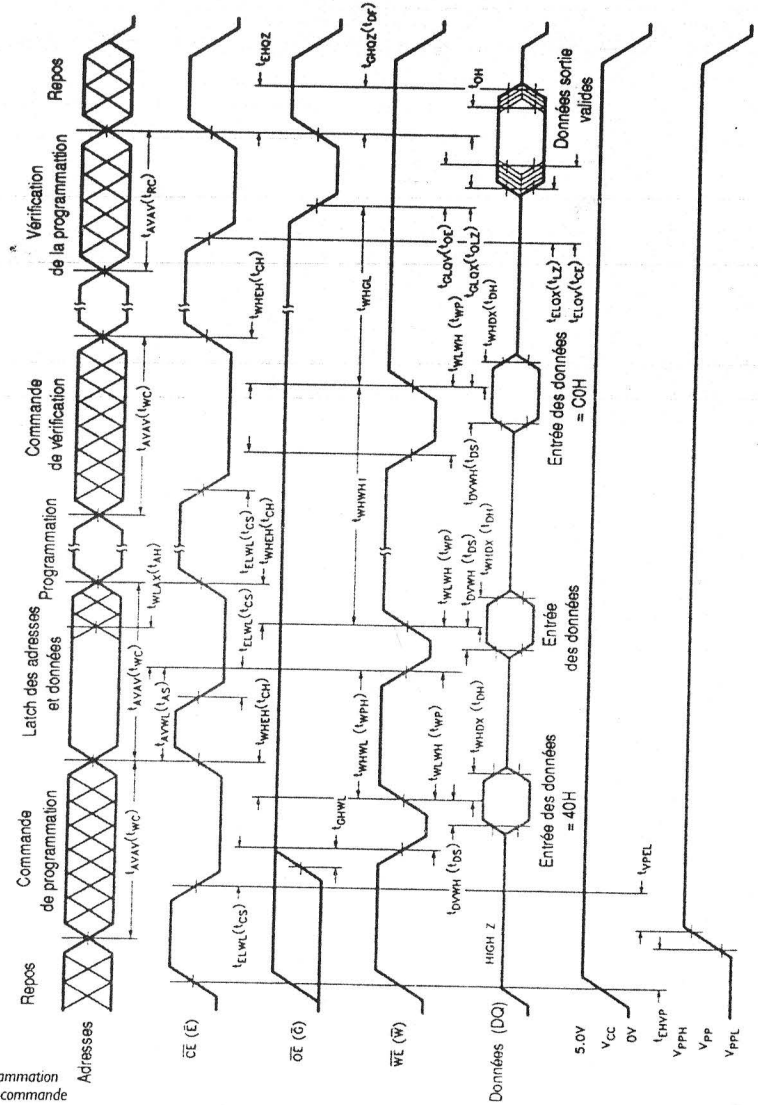


Figure 2.32 :
Chronogramme de programmation et d'accès au registre de commande d'une 28F256.

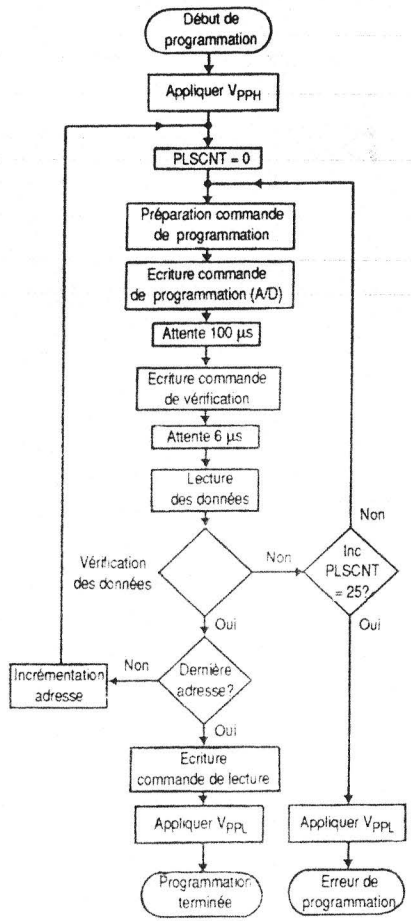


Figure 2.33 :
Algorithme de programmation d'une 28F256.

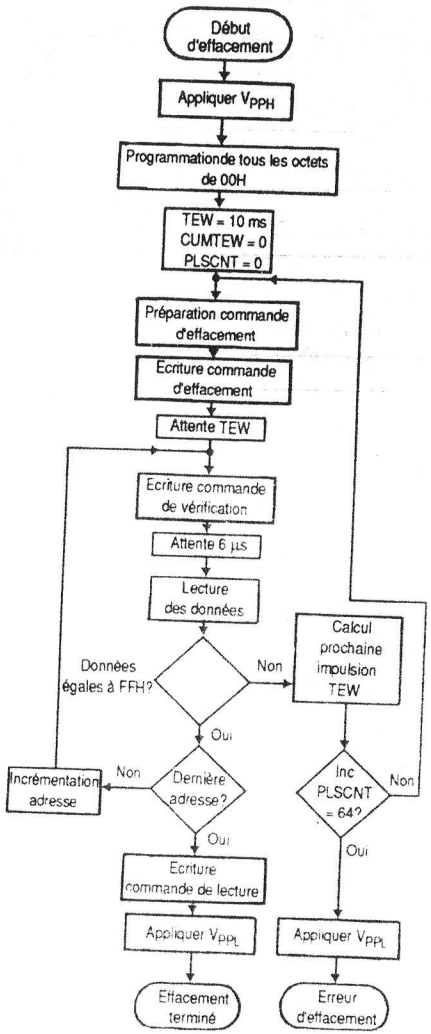


Figure 2.34 :
Algorithme d'effacement d'une 28F256.

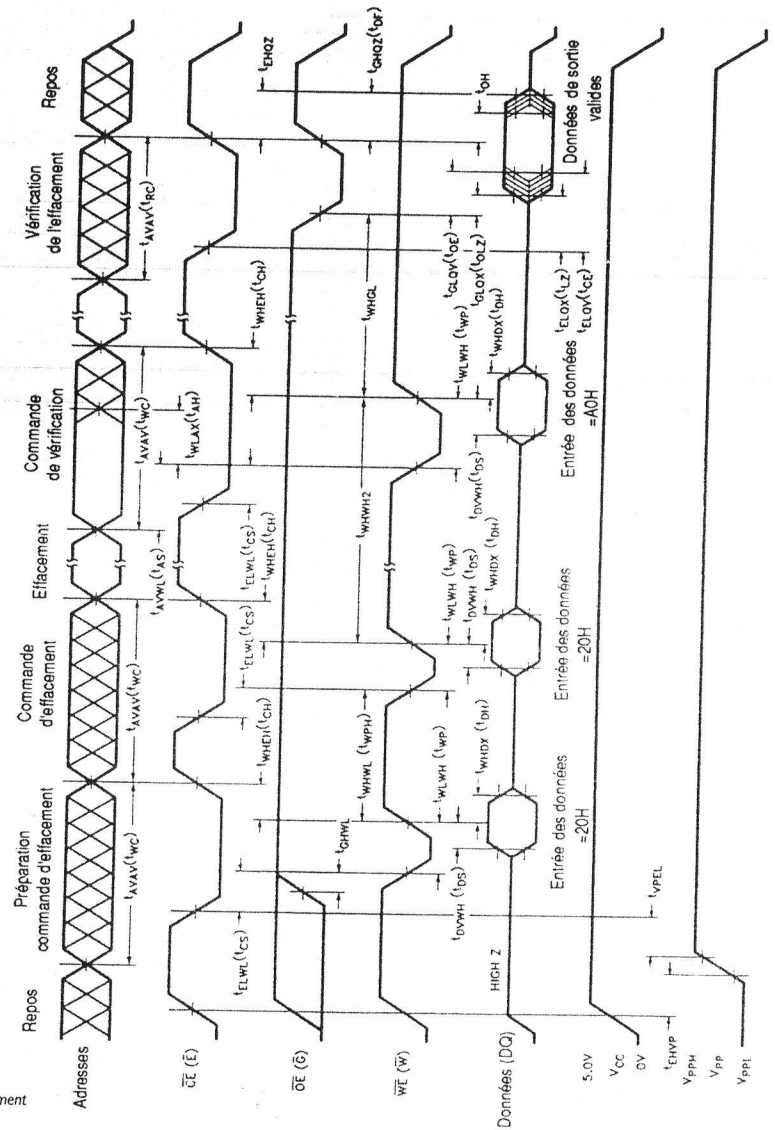


Figure 2.35 :
Chronogramme d'effacement d'une 28F256.

Les figures 2.32 et 2.35 complètent votre information en indiquant les chronogrammes associés à ces organigrammes. Leur lecture, après l'exposé sur le mode d'accès au registre de commande, ne doit vous poser aucun problème particulier.

UTILISATION DES MÉMOIRES COMME CIRCUITS LOGIQUES

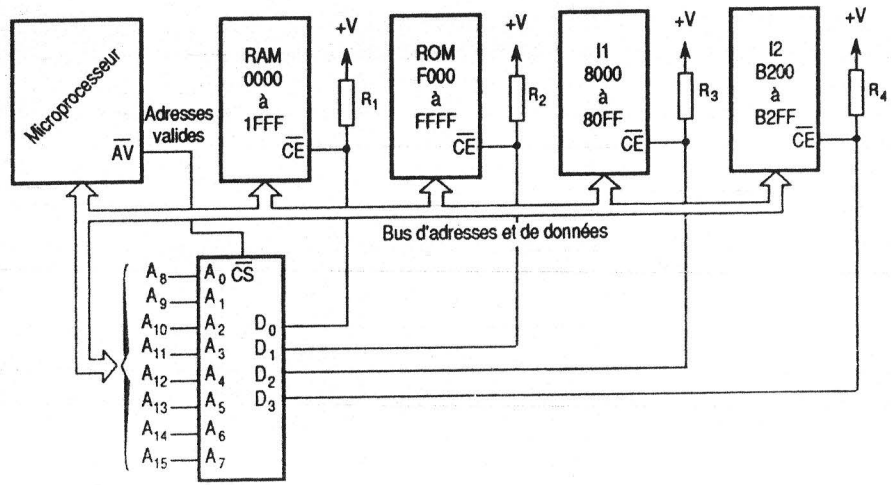
Il est évident que dans plus de 90 % des cas les mémoires, quelle que soit leur technologie, sont utilisées en tant que telles, c'est-à-dire pour stocker des données ou des programmes. On les trouve donc associées à un microprocesseur ou microcontrôleur, selon une architecture classique sur laquelle nous ne reviendrons pas dans le cadre de cet ouvrage consacré aux seuls circuits logiques programmables.

Malgré cette prédominance, les mémoires, surtout lorsqu'elles sont rapides et de petite taille, sont également utilisées comme des circuits logiques programmables en lieu et place des PAL et autres PLD dont nous parlerons dans les chapitres suivants. Les raisons d'être de ce choix sont multiples, mais on peut citer principalement : la bonne adaptation des mémoires à la réalisation de quelques fonctions logiques et le fait qu'il ne soit pas nécessaire de posséder de connaissance particulière en logique programmable, ni surtout de système de développement, pour passer à la pratique.

Bien qu'il soit possible d'envisager de nombreuses applications de ces mémoires en tant que circuits logiques programmables, nous avons choisi trois exemples, relativement simples mais tous extraits de situations réelles.

DÉCODAGE D'ADRESSES

Cette application est certainement l'une des premières à avoir utilisé des mémoires comme circuits logiques. Elle a eu beaucoup de succès lors de la commercialisation des premiers microprocesseurs 8 bits, car, à l'époque, les circuits logiques tels que les PAL et autres PLD étaient chers, assez peu répandus et ne disposaient pas de moyens de développement performants comme maintenant. De plus, l'utilisation d'une mémoire pour faire du décodage d'adresses était d'une extrême simplicité comme le montre la figure 2.36.



Adresse micro	Adresse mémoire	D0	D1	D2	D3
0000 à 1FFF	00 à 1F	0	1	1	1
2000 à 7FFF	20 à 7F	1	1	1	1
0 à 80FF	80	1	1	0	1
8100 à B1FF	81 à B1	1	1	1	1
0 à B2FF	B2	1	1	1	0
B300 à EFFF	B3 à EF	1	1	1	1
F000 à FFFF	F0 à FF	1	0	1	1

Figure 2.36 :
Utilisation d'une PROM en tant que circuit de décodage d'adresses.

Nous y voyons un microprocesseur qui doit pouvoir accéder à quatre organes différents tels que, par exemple : une RAM de données, une ROM de programme et deux circuits d'interface. Ces circuits doivent évidemment être situés à des adresses différentes dans l'espace adressable du microprocesseur. Dans cet exemple nous avons choisi :

- de F000 à FFFF pour la ROM de programme ;
- de 0000 à 1FFF pour la RAM de données ;
- de 8000 à 80FF pour le circuit d'interface numéro 1 ;
- de B200 à B2FF pour le circuit d'interface numéro 2.

Le microprocesseur dispose bien évidemment de 16 lignes d'adresses A_0 à A_{15}, et d'une ligne indiquant lorsque celles-ci sont valides que nous appellerons AV barre (active au niveau bas comme sur de très nombreux microprocesseurs 8 bits).

Sur le schéma de la figure 2.36 nous voyons une petite PROM de 256 mots de 4 bits (typiquement une PROM bipolaire à fusibles) dont le contenu est indiqué sur cette même figure.

Le fonctionnement du montage est fort simple. Lorsque cette PROM n'est pas validée c'est-à-dire lorsque la ligne AV barre est au niveau haut et que, de ce fait, les lignes d'adresses A_0 à A_{15} du microprocesseur n'ont aucune signification particulière, les sorties de la PROM sont en état haute impédance et ne valident aucun des circuits qui suivent. Toutes leurs entrées de validation, actives au niveau bas, sont en effet ramenées au niveau logique haut par les résistances R_1 à R_4.

Lorsqu'une adresse valide est présente sur le bus, la ligne AV barre passe au niveau bas, ce qui valide la PROM. Celle-ci prend en compte les adresses présentes sur A_8 à A_{15} c'est-à-dire en fait les deux chiffres de poids forts de l'adresse réelle.

Vous pouvez alors vérifier que, grâce à la programmation de la PROM réalisée par nos soins, celle-ci valide :

- la ROM de programme de F0 à FF ce qui correspond donc bien aux adresses réelles F000 à FFFF,
- la RAM de données de 00 à 01 ce qui correspond bien aux adresses réelles 0000 à 01FF,
- et ainsi de suite.

Bien sûr, la finesse de résolution de cet adressage n'est que de 256 octets puisque la PROM ne prend en compte que les 8 lignes de poids forts des adresses, mais il suffirait de mettre une mémoire de 512 mots de 4 bits pour augmenter la résolution à 128 octets, et ainsi de suite.

Un tel montage fonctionne parfaitement sous réserve, bien sûr, que le temps d'accès de la PROM soit très court. En effet, celui-ci s'ajoute au temps d'accès de l'organe adressé par le microprocesseur et il ne doit donc pas être pénalisant. Les PROM bipolaires de 256 mots de 4 bits, 32 mots de 8 bits ou bien encore 512 mots de 4 bits, avec des temps d'accès de 35 à 70 ns ont été et sont encore très utilisées dans ce genre d'application.

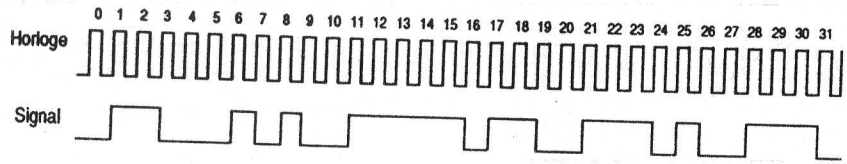
Elles font économiser un nombre important de circuits logiques (essayez, à titre d'exercice, de réaliser le décodage que nous venons de faire avec des circuits logiques classiques et vous serez convaincu).

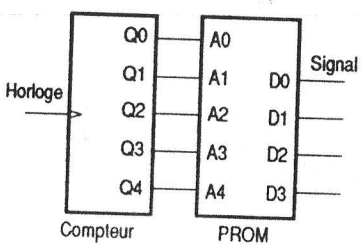
Un autre intérêt majeur de cette application est qu'elle permet très facilement de modifier l'adressage. Il suffit de remplacer la PROM par une de programmation différente. C'est très utile lorsque, par exemple, le programme de l'application grossit et qu'il faut alors utiliser une ROM de plus grande taille. Avec des circuits logiques classiques, il faudrait tout casser ; ici il suffit de remplacer un circuit.

GÉNÉRATION DE SIGNAUX COMPLEXES

La deuxième utilisation des PROM en tant que circuits logiques la plus fréquente est celle consistant à générer des signaux logiques complexes ou *irréguliers*. Examinez le signal logique dessiné à titre d'exemple figure 2.37, et essayez de le générer avec des circuits logiques classiques. Vous allez certainement y arriver en faisant suivre un compteur par de nombreuses portes mais cela va vous prendre du temps et consommer pas mal de portes. Si vous devez faire une carte avec tout cela, il vous faudra de nombreux boîtiers puisque vous n'aurez pas toutes les portes que vous désirez dans un seul ; en outre, le circuit imprimé à dessiner risque fort d'avoir un tracé assez complexe.

Nous avons représenté sur cette même figure une solution particulièrement simple faisant appel à une PROM. Un simple compteur binai-





Adresse	A_4	A_3	A_1	A_2	A_0	D_0
0	0	0	0	0	0	0
1	0	0	0	0	1	1
2	0	0	1	0	0	1
3	0	0	1	0	1	0
4	0	0	0	1	0	0
5	0	0	0	1	1	0
6	0	0	1	1	0	1
7	0	0	1	1	1	0
8	0	1	0	0	0	1
9	0	1	0	0	1	0
10	0	1	1	0	0	0
11	0	1	1	0	1	1
12	0	1	0	1	0	1
13	0	1	0	1	1	1
14	0	1	1	1	0	1
15	0	1	1	1	1	1

Adresse	A_4	A_3	A_1	A_2	A_0	D_0
16	1	0	0	0	0	0
17	1	0	0	0	1	1
18	1	0	1	0	0	1
19	1	0	1	0	1	0
20	1	0	0	1	0	0
21	1	0	0	1	1	0
22	1	0	1	1	0	1
23	1	0	1	1	1	0
24	1	1	0	0	0	1
25	1	1	0	0	1	0
26	1	1	1	0	0	0
27	1	1	1	0	1	1
28	1	1	0	1	0	1
29	1	1	0	1	1	1
30	1	1	1	1	0	1
31	1	1	1	1	1	0

Figure 2.37 :
Utilisation d'une PROM pour générer des signaux logiques complexes.

re attaque les lignes d'adresses d'une PROM imaginaire de 32 mots de 1 bit (en fait, il n'existe pas aussi petit, mais c'est pour l'exemple). Si notre mémoire contient les données indiquées dans le tableau de cette même figure et que nous attaquions notre compteur avec le signal d'horloge de la figure 2.37, vous constaterez que l'on va trouver, sur la ligne de donnée de la PROM, le signal que nous cherchons à produire. Armez-vous d'un papier et d'un crayon pour le vérifier si nécessaire. Et si d'aventure vous avez oublié un créneau quelque part sur le signal, il suffira de programmer différemment la PROM pour y remédier.

AUTOMATE PROGRAMMÉ SIMPLIFIÉ

Dans de nombreuses situations telles que la commande d'automatismes simples (machines-outils, appareils ménagers, jeux de lumière professionnels), une PROM permet de réaliser des miracles en utilisant des schémas d'une extrême simplicité.

La figure 2.38 donne le schéma, partiellement synoptique, d'un tel ensemble. Un oscillateur, à fréquence fixe ou variable, attaque un compteur dont les sorties commandent à leur tour les lignes d'adresses d'une mémoire. Les lignes de données de celle-ci, via des interfaces de puissance (relais, triacs, etc.) commandent ensuite les divers éléments de l'application (moteurs, électrovannes, ampoules).

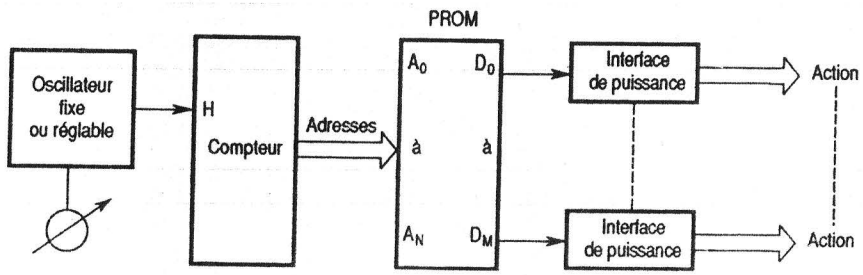


Figure 2.38 :
Principe d'un automate programmable simplifié à PROM.

Pour peu que la mémoire soit correctement programmée, il est donc possible de définir très exactement les cycles de marche et d'arrêt de ces différents organes commandés. La vitesse de changement d'état et donc, la durée des cycles ne dépend que de la fréquence de l'horloge qui pilote le compteur des adresses.

Les temps d'accès n'étant généralement pas critiques dans une telle application, l'utilisation d'une UVPROM de grande capacité est parfaitement réaliste et permet de disposer de nombreux pas de programmes voir même de disposer de plusieurs programmes différents, que l'on sélectionne par exemple par commutation des lignes d'adresses de poids forts.

CONCLUSION

Nous en resterons là de ces exemples d'utilisation des mémoires en tant que circuits logiques, car ce sont à peu près les seuls que l'on rencontre encore actuellement. En effet, les *vrais* circuits logiques programmables ont fait tellement de progrès et sont devenus tellement peu coûteux qu'on y fait maintenant appel sans hésiter. Bien sûr, il faut savoir lequel choisir et comment le mettre en œuvre en fonction du problème à résoudre. La suite de cet ouvrage est là pour vous guider dans ce choix parfois délicat.

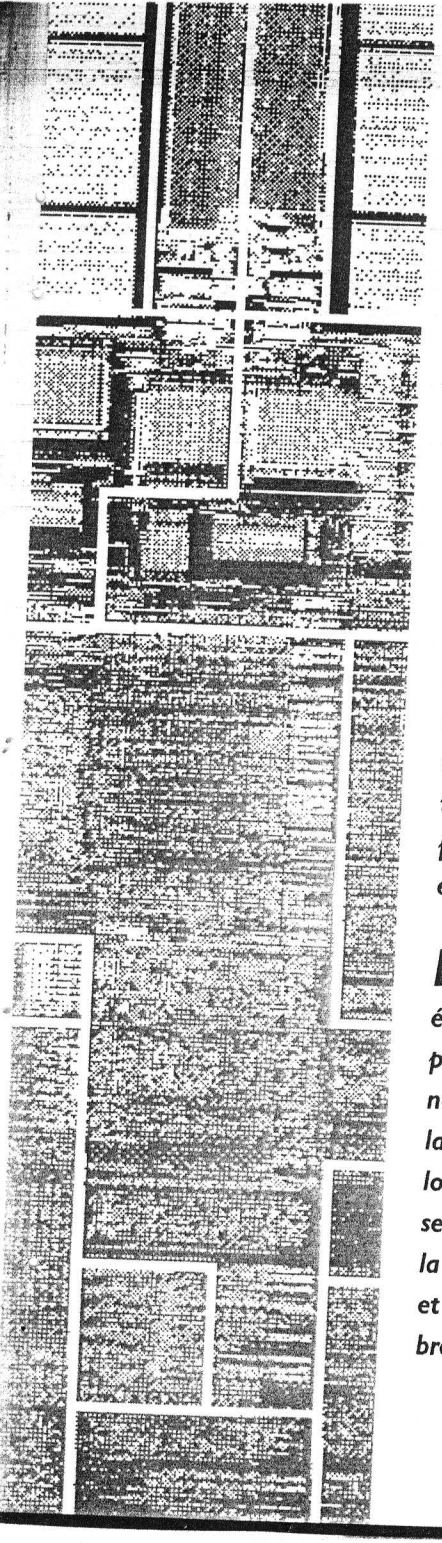
CIRCUITS LOGIQUES PROGRAMMABLES

CHAPITRE 3

Les PAL bipolaires.

PAL, PAL bipolaires, PAL versatiles, VPAL

- Notions de base 79
- Les PAL combinatoires et à registres classiques 94
- Les PAL versatiles ou VPAL . 105
- Un exemple d'application ... 110



Bien que le terme PAL soit en fait une marque déposée de MMI (qui a été absorbé depuis par AMD) ; les PAL ont eu tellement de succès dès leur commercialisation que cette appellation est passée dans le langage courant. C'est tout à la fois réconfortant car cela prouve que le concept était vraiment utile, mais aussi gênant car ce terme est bien souvent utilisé à tort et à travers pour désigner des circuits aux contenus parfois bien différents.

Nous allons voir dans ce chapitre les concepts de base des PAL ainsi que les conventions de notation et d'utilisation qui les caractérisent, puis nous ferons un tour d'horizon des différentes familles de produits disponibles sur le marché et nous terminerons par un exemple d'application.

Les PAL CMOS ou E2PAL ou encore PAL EECMOS étant assimilables aux GAL, ils sont traités dans le chapitre suivant, intégralement consacré à ces circuits. Nous ne parlerons donc, dans ce chapitre, que des PAL bipolaires ou PAL à fusibles ; néanmoins nous vous conseillons fortement sa lecture, même si ces circuits ne vous semblent d'aucun intérêt. Ils sont en effet à la base de la majorité des circuits logiques programmables actuels, et les concepts présentés ici se retrouvent dans de nombreuses autres familles de circuits plus complexes.

NOTIONS DE BASE

Comme nous l'avons dit, le concept de PAL a été inventé il y a maintenant plus de 20 ans par la société MMI qui était, à l'époque, un des pionniers de la technologie des mémoires PROM à fusibles dont nous avons largement parlé au chapitre précédent.

L'idée de départ a été relativement simple puisque les ingénieurs de MMI se sont dit qu'il serait intéressant d'utiliser la technologie des fusibles que l'on sait faire *sauter* de l'extérieur pour pouvoir réaliser des circuits logiques dont les interconnexions internes seraient ainsi rendues programmables.

Comme les fusibles ne peuvent que couper des liaisons lors de leur destruction, il a fallu trouver une méthode de réalisation des circuits propre à s'accommoder de cette limitation. Ainsi est née la structure présentée figure 3.1 qui est le fondement des PAL.

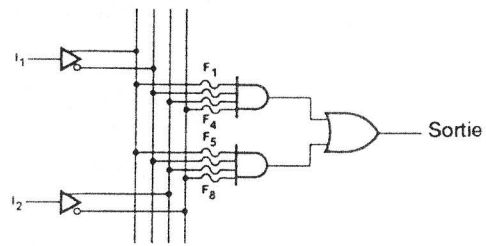


Figure 3.1 :
Le principe de base d'un PAL

Présenté tel quel, ce montage comporte deux entrées I_1 et I_2 et une sortie O. Un certain nombre de fusibles (huit dans le cas présent) permettent de réaliser de nombreuses fonctions logiques diverses. Lors de la livraison du PAL, tous ces fusibles sont intacts et la programmation va donc consister à faire sauter les fusibles nécessaires afin de réaliser la fonction de son choix. On le voit, cela ressemble comme deux gouttes d'eau à ce que l'on fait pour une PROM bipolaire à fusibles.

CONVENTIONS DE NOTATION

Bien, qu'elle rende fidèlement compte de la réalité, la représentation adoptée figure 3.1 n'est pas vraiment utilisable en pratique. En effet, elle occupe beaucoup de place et conduit vite à des schémas illisibles

surtout lorsque l'on sait que les PAL les plus courants contiennent plusieurs dizaines d'exemplaires de cette cellule élémentaire.

Une autre représentation schématique, visible figure 3.2, a donc été adoptée dans l'industrie. Cette figure permet de comprendre sa genèse et la logique de son évolution. En *a* nous voyons la notation logique habituelle pour une porte ET à 3 entrées prise en exemple. Elle se transforme tout naturellement en ce que nous pouvons voir en *b* dans le cas d'un PAL où les croix représentent les fusibles intacts. Dès lors, le schéma de notre figure 3.1 peut se transformer en ce que nous vous proposons en 3.2 *c*. C'est tout aussi facile à lire mais déjà nettement moins encombrant.

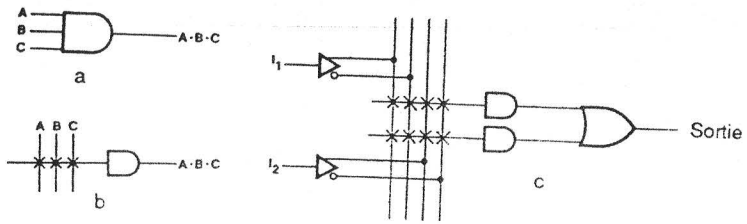


Figure 3.2 :
La génèse de la "notation PAL".

Voyons maintenant une utilisation pratique de ce genre de représentation avec la réalisation d'une équation logique simple : un banal OU exclusif. Vous savez tous que le OU exclusif de deux signaux que nous appellerons I_1 et I_2 peut être représenté par :

$$I_1 + I_2 = I_1.\overline{I_2} + \overline{I_1}.I_2$$

Une telle équation se réalise, en logique câblée classique, comme indiqué figure 3.3. Jusque là, il n'y a rien que de très ordinaire.

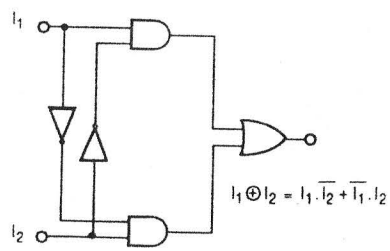


Figure 3.3 :
Un OU EXCLUSIF en notation logique classique.

La réalisation d'une telle fonction avec un PAL est tout aussi simple comme le montre la figure 3.4. Nous y retrouvons en effet notre cellule de base dans laquelle quatre fusibles ont été détruits (les endroits où les croix sont absentes), et nous constatons que cela réalise bien l'équation logique attendue.

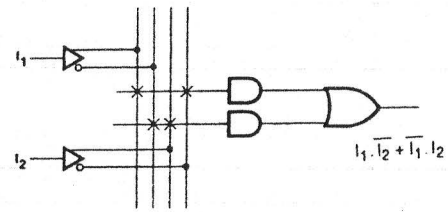


Figure 3.4 :
Le même OU EXCLUSIF réalisé dans un PAL.

LES DIVERSES STRUCTURES DISPONIBLES

L'exemple que nous avons choisi n'était pas pris tout à fait au hasard il faut bien le reconnaître, et se prêtait merveilleusement bien à une réalisation avec la cellule de base que nous vous avions présentée. Ce n'est hélas pas toujours le cas dans la réalité, ce qui nous amène à réfléchir un peu aux différentes structures de circuits logiques qu'il serait souhaitable de trouver sur le marché pour pouvoir réaliser facilement toutes les équations logiques possibles.

En utilisant la notation vue précédemment, vous trouverez en figure 3.5 la structure mémoire d'une PROM bipolaire à fusibles. Ainsi que nous l'avions indiqué au chapitre 2 et comme vous pouvez le vérifier facilement sur cette figure, une telle mémoire est en fait l'association d'une zone de ET fixes, constituée par la partie décodage d'adresses, et d'une zone de OU programmables constituée par le plan mémoire proprement dit.

Cela permet de comprendre que, outre le stockage de données qui est sa fonction première, une mémoire puisse facilement être utilisée en tant que circuit logique comme nous l'avons vu grâce aux quelques exemples donnés en fin de chapitre 2.

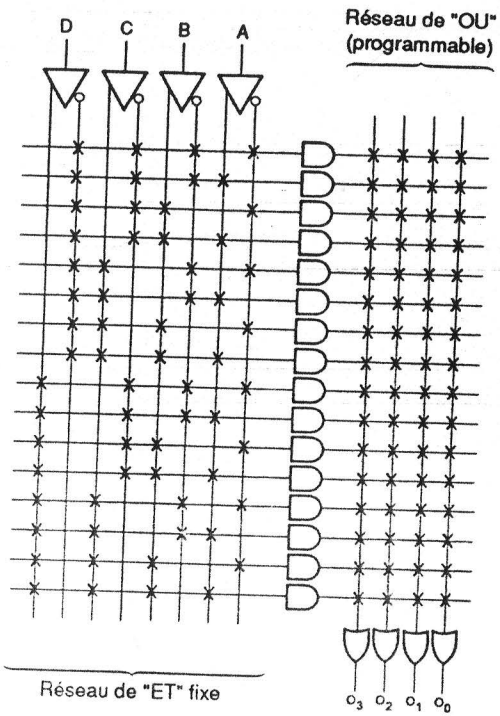


Figure 3.5 :
Structure logique d'une PROM à fusibles.

Un PAL *vrai* (par vrai, nous entendons ce qui avait initialement été appelé PAL par leur concepteur MMI) a une structure analogue à celle présentée figure 3.6. Les similitudes avec la PROM sont évidentes, mais ici, la zone de ET d'entrées est programmable, alors que la zone de OU de sorties est fixe. En première approximation on pourrait croire que la souplesse d'utilisation d'un PAL est analogue à celle d'une PROM puisque l'on observe une simple permutation des zones programmables. En fait, comme nous le verrons, il n'en est rien grâce à divers artifices.

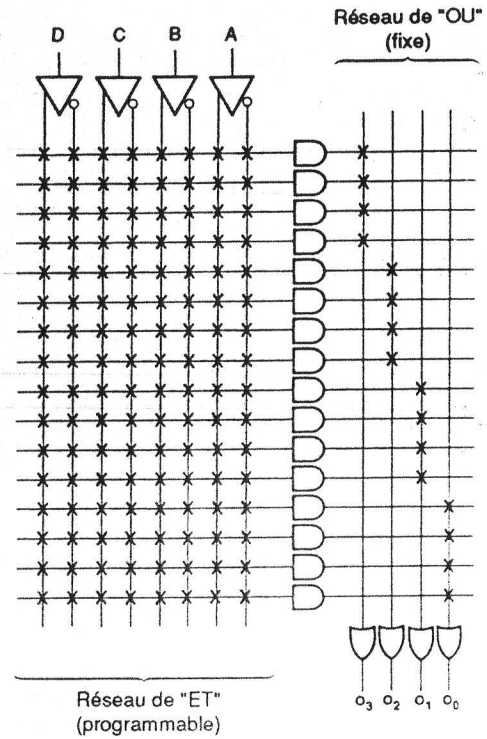


Figure 3.6 :
Structure logique d'un PAL

Poursuivant notre présentation des structures internes, nous sommes naturellement conduit à vous proposer celle de la figure 3.7 dans laquelle les deux zones, ET et OU, ont été rendues programmables. Les circuits de ce type sont évidemment très souples d'emploi, mais si on veut les exploiter vraiment à fond, ils demandent plus de réflexion que les PROM ou les PAL. Leur appellation officielle théorique initiale était FPLA, ce qui signifiait mot-à-mot *Field Programmable Logic Array*, c'est-à-dire réseau logique programmable sur site ; la différence avec la signification du terme PAL n'est pas évidente mais nous n'y pouvons rien !

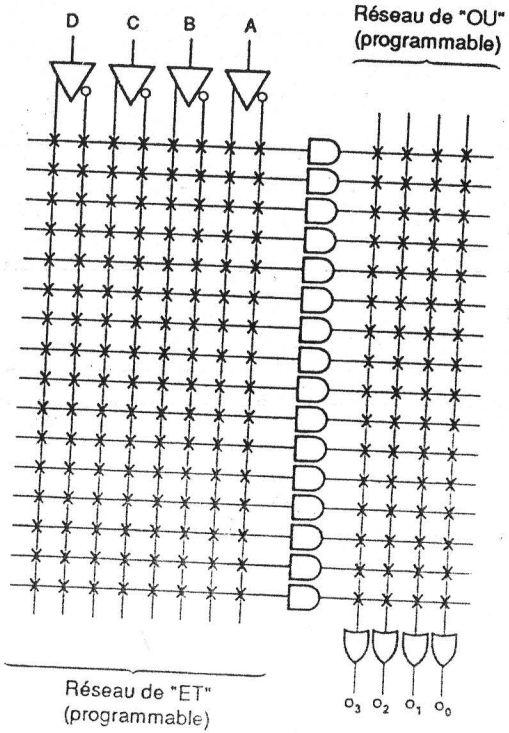


Figure 3.7 :
Structure logique d'un FPLA.

Les trois structures que nous venons de voir ne font appel qu'à de la logique combinatoire. Pour introduire des notions de temps et donc, faire de la logique séquentielle, il faut ajouter des registres à nos schémas. C'est possible dans le cas des PROM, des PAL et des FPLA.

LES ENTRÉES/SORTIES DES PAL

Maintenant que nous avons une idée un peu plus précise des diverses structures internes qu'il est possible de rencontrer, examinons plus en détail les éléments accessibles de l'extérieur, à savoir les pattes d'entrées ou de sorties à notre disposition.

Tous les PAL, sans exception, disposent d'un certain nombre d'entrées qui aboutissent toutes, sous forme directe et inversée, sur la matrice de fusibles de programmation. La figure 3.6 reste sur ce plan parfaitement correcte. Les sorties, par contre, peuvent revêtir plusieurs aspects. Les plus simples sont conformes à ce que nous avons représenté figure 3.8.

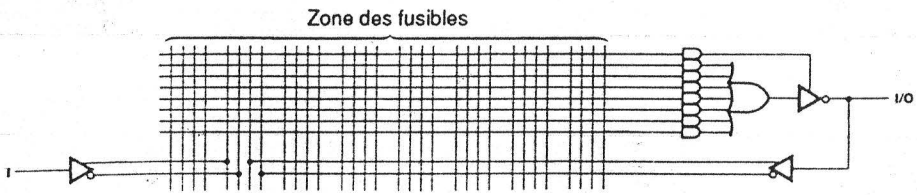


Figure 3.8 :
Les entrées/sorties d'un PAL combinatoire.

Le signal résultant de la zone de fusibles programmables est appliqué à la patte de sortie, ou plus exactement d'entrée/sortie via un buffer trois états, lui-même commandé par cette même zone de fusibles. Cela signifie donc que l'état de la sortie : valide ou haute impédance, pourra être une fonction logique des signaux d'entrée de votre choix.

Cette même patte est reliée, comme les pattes d'entrées, à un inverseur/non inverseur qui attaque à son tour la zone de fusibles programmables. Il est ainsi possible :

- soit de ré-injecter le signal de sortie dans la zone d'entrée ;
- soit d'utiliser tour à tour la patte de sortie comme patte d'entrée en faisant passer le buffer de sortie dans le troisième état.

On le voit, une telle approche est particulièrement souple. Elle se rencontre sur tous les PAL, même les plus simples. Seuls varient d'un PAL à l'autre, comme nous le verrons tout à l'heure, le nombre d'entrées et le nombre d'entrées/sorties.

Les PAL à registres utilisent une approche un petit peu différente comme le montre la figure 3.9. En effet, la sortie du réseau de fusibles programmables aboutit sur l'entrée d'une bascule D (le registre de sortie). La sortie Q barre de ce registre est ré-injectée sur la matrice de fusibles via un inverseur/non inverseur comme dans le cas des

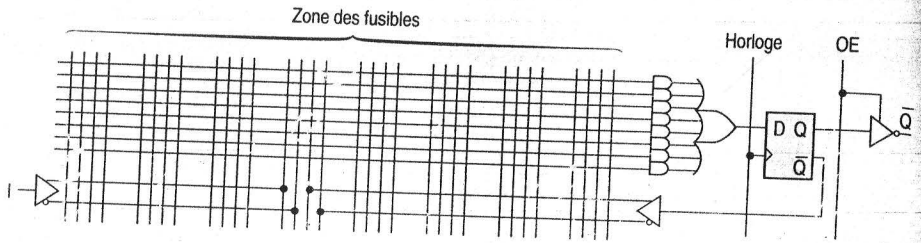


Figure 3.9 :
Les entrées/sorties d'un PAL à registres.

PAL simples. La sortie Q de ce même registre est par contre envoyée sur la patte de sortie via un buffer trois états, mais, contrairement aux PAL simples, cette patte de sortie ne peut être utilisée en entrée car elle ne dispose d'aucun accès direct vers la matrice de fusibles.

Remarquez en outre que les entrées d'horloge des bascules sont communes pour toutes les sorties et sont donc commandées par une seule et unique ligne appelée CLOCK. De même, les buffers de sorties ne sont plus validés individuellement, mais sont sous le contrôle d'une ligne commune appelée OE (*output enable ou validation des sorties*).

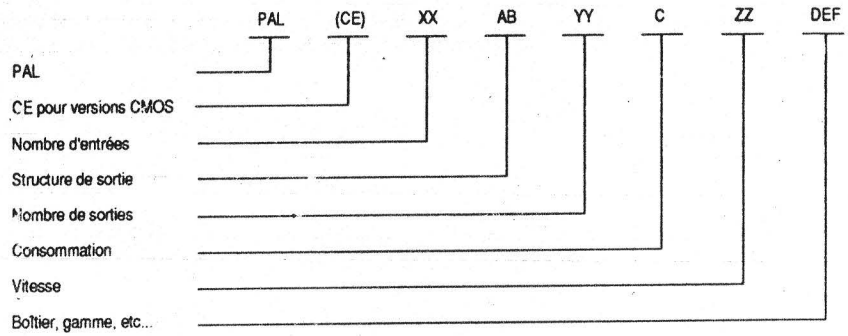
Ces deux configurations, PAL simples et PAL à registres, sont les plus répandues sur le marché si l'on excepte les PAL universels ou versatiles dont nous parlerons en fin de chapitre.

LE DÉCODAGE DES RÉFÉRENCES

Ces diverses possibilités, et une relative standardisation des produits proposés sur le marché, ont conduit les fabricants de PAL à adopter une nomenclature sinon parfaite, du moins relativement facile à décoder une fois que l'on en possède les clefs. Nous allons vous présenter celles-ci, maintenant, pour deux fabricants importants : AMD, qui est certainement le leader mondial des PAL de tous types, et Texas Instruments qui le suit de près. Avec ces deux informations vous serez à même de lire toutes les références de PAL logiques actuelles.

La figure 3.10 présente le mode de marquage AMD. Nous voyons que la référence est décomposée en un certain nombre d'éléments significatifs :

- Un terme PAL qui précise le type de produit.



Lettre(s) code(s)	Structure de sortie
L	Combinatoire active bas
H	Combinatoire active haut
R	Registre
RA	Registre asynchrone
X	Registre et OU exclusif
V	Versatile

Figure 3.10 :
Principe de marquage des PAL chez AMD.

- Un terme CE, qui est présent dans le cas des PAL CMOS effaçables électriquement et absent dans le cas des PAL bipolaires.
- Un nombre qui indique le nombre d'entrées disponibles et qui se retrouve sur tous les marquages de PAL quelle qu'en soit la marque.
- Une ou deux lettres qui indiquent le type de structure de sortie conformément au tableau de cette même figure 3.10 qui lui aussi, est quasiment commun à tous les fabricants.
- Un nombre qui indique le nombre de sorties disponibles. Ces dernières peuvent être des sorties simples ou des entrées/sorties. Ici encore, ce nombre se retrouve chez tous les fabricants.
- Une lettre qui indique le type de PAL pour ce qui est de la consommation (normale, faible ou très faible).
- Un nombre qui indique la vitesse du PAL ou plus précisément son temps de propagation typique par porte en nanosecondes.

- Diverses autres lettres qui, comme dans le cas des circuits intégrés *ordinaires*, codent le type de boîtier utilisé et la gamme de température de fonctionnement du circuit.

Muni de ces informations et d'un PAL type PAL16R8H15PC vous savez immédiatement que vous êtes en présence :

– d'un PAL bipolaire à cause du sigle,

– qu'il dispose de 16 entrées à cause du nombre 16,

– que c'est un PAL à registres grâce à la lettre R,

– qu'il dispose de 8 sorties à cause du chiffre 8,

– que c'est un modèle demi-puissance grâce à la lettre H,

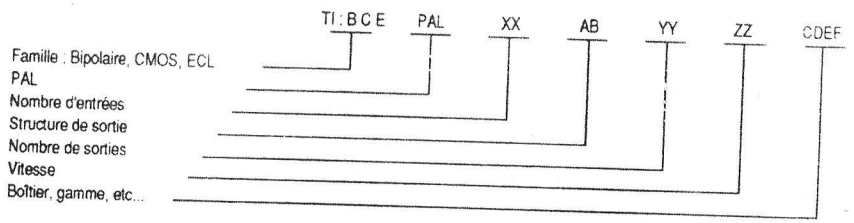
– qu'il est en boîtier DIL plastique et en gamme de température civile grâce aux lettres P et C.

Chose promise chose due, voici maintenant la clef utilisée par Texas Instruments qui ressemble bien évidemment à celle d'AMD, étant donné que les informations à fournir sont du même type.

La figure 3.11 présente le mode de marquage utilisé sur lequel on rencontre les éléments suivants :

- Trois lettres qui indiquent la famille technologique de PAL : bipolaire, CMOS ou ECL.
- Trois autres lettres qui indiquent le type de PAL selon une classification propre à Texas.
- Un nombre qui indique le nombre d'entrées conformément à l'universalité du mode de marquage dont nous avons parlé précédemment.

Figure 3.11 :
Principe de marquage des PAL chez Texas.



- Une ou deux lettres qui indiquent le type de structure de sortie comme indiqué précédemment.
- Un nombre qui indique le nombre de sorties, toujours conforme à ce que nous avons déjà vu.
- Un tiret suivi d'un nombre qui indique le temps de propagation typique par porte.
- Une ou plusieurs lettres qui codent le type de boîtier et la gamme de température d'utilisation.

Ainsi, un PAL Texas Instruments qui s'appelle TIBPAL16R8-15CN est-il strictement équivalent au PAL AMD choisi en exemple. Cela se voit très bien à la *racine* de la notation dans laquelle on retrouve les termes PAL16R8. Les autres lettres ne servent qu'à répertorier boîtier et gamme de température qui différent bien souvent d'un fabricant à un autre.

PAL BIPOLAIRES, PAL CMOS ET GAL

Si à l'origine, et comme nous venons de l'expliquer, tous les PAL étaient bipolaires puisqu'ils utilisaient la même technologie que les PROM à fusibles, il existe maintenant des PAL CMOS faisant appel aux mêmes technologies que les mémoires EEPROM vues au chapitre précédent.

De ce fait, ces PAL CMOS sont programmables et effaçables électriquement, ce qui présente évidemment un intérêt certain pour les phases d'études de produits, mais aussi en production de série car leur complexité arrive à être supérieure à celle des PAL bipolaires.

Comme pour les mémoires, cette utilisation de la technologie CMOS ne se fait pas au détriment de la vitesse et, bien que quelques PAL bipolaires arrivent à être un tout petit peu plus rapides que les PAL CMOS, cela n'est sensible que pour des applications critiques.

Compte tenu de ces différences technologiques, on retrouve les mêmes distinctions qu'avec les mémoires. Les PAL bipolaires, à fusibles donc, ne sont programmables qu'une fois, puisqu'il y a rupture physique d'une connexion. La durée de rétention de l'information est donc illimitée dans le temps si la programmation a été réalisée sur

un appareil agréé par le fabricant des PAL. Aucune possibilité d'effacement n'est évidemment prévue et les seules modifications que l'on peut faire sur un PAL déjà programmé consistent à détruire un fusible qui ne l'aurait pas été, et c'est tout.

Comme les PROM bipolaires, les PAL bipolaires ne peuvent être testés à 100 % quant à leur programmabilité, et les fabricants échangent donc gratuitement les circuits qui ne seraient pas correctement programmables.

Pour importantes qu'elles puissent être, ces limitations doivent être minimisées au stade de la production de série d'un produit. En effet, dans ce cas le contenu du PAL est parfaitement arrêté et l'absence de possibilité de re-programmation n'est pas forcément gênant.

En ce qui concerne la phase de développement d'un produit, le point de vue est totalement différent et, jusqu'à l'avènement des PAL CMOS ou GAL, ou d'autres familles technologiques présentées ci-après, l'utilisation de PAL était coûteuse en raison de l'impossibilité dans laquelle on se trouvait de modifier un circuit mal programmé. L'avènement des PAL CMOS a considérablement modifié cette façon de voir les choses, et ceux-ci sont maintenant largement utilisés lors de la mise au point d'un produit.

Après une phase relativement incertaine pendant laquelle des produits divers ont existé sur le marché, les familles de PAL CMOS sont maintenant bien stabilisées et sont normalisées chez la majorité des fabricants. Comme en outre cette technologie s'apparente à celle des GAL, nous traitons donc ces PAL CMOS au chapitre suivant, en même temps que les GAL avec lesquels ils sont au minimum fonctionnellement compatibles.

Qu'il soit bipolaire ou CMOS, un PAL se programme avec un outil approprié qui peut être un appareil autonome ou une carte placée dans un micro-ordinateur. Même s'il existe quelques schémas de programmateurs simples, ceux-ci ne peuvent guère être utilisés que pour de toutes petites réalisations ou pour de la formation en école ou lycée technique. En effet, outre le problème de programmation proprement dit qui, lui, reste relativement facile à résoudre car proche de celui utilisé pour les PROM bipolaires, se pose le problème du développement du PAL. Un programme spécifique est alors nécessaire.

COPIE DE PAL ET FUSIBLE DE SÉCURITÉ

Le développement d'une application comprenant un ou plusieurs PAL nécessite généralement un travail assez long, comme le développement de tout montage faisant appel à de la logique d'ailleurs mais, du fait de la mise en œuvre très facile qui en résulte ensuite, puisque très peu de boîtiers et un circuit imprimé au tracé relativement simple sont nécessaires ; il arrive que des entreprises indélicates soient tentées de copier les PAL développés par leurs concurrents.

En effet, une fois placé sur un programmateur, un PAL se comporte un peu comme une mémoire qu'il est possible de programmer, bien sûr, mais aussi de relire. Rien ne semble alors pouvoir empêcher le concurrent indélicat de copier le fruit de vôtre travail. Pour remédier à cela, et dès l'origine des PAL, les fabricants ont donc prévu la mise en place d'un fusible ou bit de sécurité le *security bit* ou *security fuse* que l'on peut ou non décider de faire sauter à tout instant.

Une fois qu'il est détruit, ce fusible interdit toute relecture du PAL par quelque méthode que ce soit. De ce fait, la procédure d'utilisation de ce bit est généralement la suivante.

Lorsque l'on programme le PAL, on laisse ce fusible intact. On relit le PAL pour s'assurer qu'il est correctement programmé et, après l'avoir éventuellement essayé une dernière fois sur l'application, on fait sauter ce fusible. Le PAL est alors définitivement protégé contre la copie.

Bien sûr, nous direz-vous, il reste toujours la possibilité d'appliquer des stimuli sur les pattes du circuit et d'observer le comportement de ses sorties, mais si cette démarche reste faisable pour des PAL combinatoires très simples, elle est sans intérêt pour des PAL complexes ou séquentiels car on a alors plus vite fait de concevoir soi-même un PAL que de déterminer par expérimentation le contenu du PAL à copier. La protection apportée par ce fusible de sécurité peut ainsi être qualifiée d'excellente.

Précisons dès à présent que les PAL CMOS, bien qu'ils soient effaçables électriquement, disposent évidemment eux aussi de ce fusible de sécurité. Comme pour les PAL bipolaires, il interdit toute relecture du PAL et, afin qu'il apporte une réelle protection, il ne peut être effacé seul mais uniquement lors d'un effacement global du PAL.

LES OUTILS DE DÉVELOPPEMENT

Lors de la commercialisation des premiers PAL, aucun outil informatique n'existait pour assister le concepteur qui réalisait son développement à la main. En d'autres termes, après avoir conçu son schéma logique, sous forme de tables de vérité, de portes ou bien encore d'équations, il fallait qu'il détermine la table des fusibles à programmer dans le PAL pour réaliser ce schéma.

Si une telle démarche reste encore acceptable aujourd'hui pour des circuits simples, elle devient bien vite inextricable pour des circuits complexes et source de nombreuses erreurs. Des logiciels, de plus en plus performants, ont donc été commercialisés par, ou sous l'impulsion des fabricants de PAL dont le plus célèbre est très certainement PALASM de la défunte société MMI.

Ce logiciel qui, dans sa version de base, ne fait appel à aucune possibilité graphique et peut, de ce fait, fonctionner même sur de très petits systèmes, permet de définir les affectations des broches d'un PAL ainsi que les équations logiques des fonctions à réaliser. La notation utilisée est naturelle et ne déroute pas l'utilisateur après un apprentissage de quelques minutes.

Bien sûr, depuis ses débuts, ce logiciel a considérablement évolué pour profiter au mieux des possibilités des micro-ordinateurs actuels.

Malgré son succès initial et son ancienneté, PALASM n'est plus seul sur le marché depuis déjà longtemps. Le tableau de la figure 3.12 donne ainsi quelques références de produits commercialisés pour le développement des PAL, mais cette liste n'est pas limitative.

Ce qu'il est important de noter, c'est que quasiment tous les logiciels de conception de PAL actuels comportent maintenant une fonction de simulation. Grâce à cela, on peut appliquer des signaux logiques au PAL que l'on se propose de programmer et vérifier son comportement.

Même si cela peut vous sembler secondaire avec des PAL CMOS effaçables, détrompez-vous tout de suite. En effet, il est bien évidemment impossible d'accéder aux nœuds de connexions internes d'un PAL programmé, à moins, bien sûr, d'avoir prévu de faire sortir ceux-ci sur des pattes, mais on est bien vite limité en ce domaine. Si le PAL

Logiciel	Fournisseur
PALASM® 90 Software (PAL and PLS devices) Am29CPL100 Software (Am29CPL151/4) PEGPDS™ (Am2971) Programmable Gate Array Software (Am2000/3000 Series)	Advanced Micro Devices Contact local sales office
ABEL™ and PLDlinx™ FutureDesigner™ PLDtest® and PLDtest Plus™ (Vector Generation)	Data I/O Corporation 10525 Willows Road N.E. P.O. Box 97046 Redmond, WA 98073-9746 (800) 247-5700 or (206) 881-6444
CUPL™	Logical Devices 1201 E. Northwest 65th Place Fort Lauderdale, FL 33309 (800) 331-7766 or (305) 491-7405
LOG/iC®	ISDATA 800 Airport Road Monterey, CA 93940 (408) 373-7359
PLDesigner®	Minc Inc. 6755 Earl Drive, Suite 200 Colorado Springs, CO 80918 (719) 590-1155
OrCAD/PLD™	OrCAD Systems Corporation 1049 S.W. Baseline Street, Suite 500 Hillsboro, OR 97123 (503) 640-9488
Anvil ATG™ (Vector Generation)	Anvil Software, Inc. 427-3 Amherst Street, Suite 391 Nashua, NH 03063 (603) 891-1995
Test Generator (Vector Generation)	ATG Associates 3415 Merrill Road Aptos, CA 95003 (408) 475-5717

Figure 3.12 :
Quelques logiciels de développement et de programmation de PAL.

ne fonctionne pas comme prévu et que sa conception soit assez complexe avec, par exemple, de nombreux rebouclages, il peut être difficile de déceler une erreur.

Avec un simulateur bien conçu, il est possible de placer des sondes à l'intérieur du PAL, puisque celui-ci n'a aucune réalité physique, et de vérifier ainsi son comportement étape par étape. La mise au point s'en trouve considérablement facilitée.

Précisons, avant d'en terminer avec ce sujet, que les simulateurs sont des produits relativement coûteux, et ce, d'autant plus qu'ils sont performants. Certains produits font en effet une simple simulation fonctionnelle du PAL ce qui peut être suffisant pour certaines applications. D'autres font une simulation fonctionnelle et temporelle, c'est-à-dire qu'ils savent prendre en compte les temps de propagation internes. La qualité de la simulation réalisée est alors excellente mais le prix du produit s'en ressent évidemment !

LES PAL COMBINATOIRES ET À REGISTRES CLASSIQUES

Ces PAL sont évidemment les plus anciens de la famille et, si vous avez lu ce qui précède, vous savez déjà tout ou presque à leur sujet. Leur architecture est en effet conforme à ce que nous venons de vous présenter. Il existe de nombreuses références, construites comme nous venons de l'expliquer, qui permettent de faire varier le nombre d'entrées et le nombre de sorties ainsi que les structures de ces dernières. Ces diverses familles de référence conduisent à des circuits qui, lorsqu'ils sont fournis en boîtiers DIL, sont logés dans des modèles 20 pattes ou 24 pattes. On parle alors couramment de la famille des PAL 20 pattes ou de la famille des PAL 24 pattes.

Si, par le passé, une multitude de références diverses avec des configurations d'entrées/sorties très riches ont été proposées, on ne trouve plus aujourd'hui que deux familles principales : la famille des PAL 20 pattes ou famille 16R8 et la famille des PAL 24 pattes ou famille 20R8.

La famille des PAL 16R8 comporte quatre membres dont les principales caractéristiques sont résumées figure 3.13. Ces circuits sont organisés comme le montrent les figures 3.14 et 3.15. Remarquez en

Référence	Entrées dédiées	Sorties	Termes produits par sortie	Retour	Validation des sorties
16L8	10	6 combinatoires 2 combinatoires	7 7	E/S –	Programmable Programmable
16R8	8	8 à registres	8	Registres	Patte
16R6	8	6 à registres 2 combinatoires	8 7	Registres E/S	Patte Programmable
16R4	8	4 à registres 4 combinatoires	8 7	Registres E/S	Patte Programmable

Figure 3.13 :
Les membres de la famille des PAL 20 pattes 16R8 .

particulier les variantes au niveau de la possibilité de commande de la fonction validation des sorties (OE) qui est programmable sur certains boîtiers et accessible sur une patte pour d'autres.

Le brochage adopté par ces circuits est rappelé figure 3.16 tandis que la figure 3.17 donne une vue complète d'un 16R8, équivalente au bas de la figure 3.15.

Ces PAL existent dans diverses gammes de vitesse déterminées par le suffixe placé après leur référence. Afin que vous ayiez une idée précise de ce que l'on peut en attendre, la figure 3.18 vous donne quelques chiffres que nous vous laissons le soin d'apprécier. Remarquez en particulier le temps t_{PD} qui mesure le délai entre une entrée et une sortie combinatoire. Il peut descendre à 1 ns pour la version la plus rapide de ces circuits !

La deuxième famille de PAL ou PAL 24 pattes est la famille 20R8 qui comporte quatre références différentes comme indiqué figure 3.19. L'organisation interne de ces circuits est analogue à celle déjà vue pour les PAL 20 pattes comme le montrent les figures 3.20 et 3.21.

Le brochage, quant à lui, vous est présenté figure 3.22 pour les différentes configurations de boîtiers disponibles. La figure 3.23 montre l'organisation interne détaillée d'un 20R8 qui vous permet de mieux comprendre la signification du schéma du bas de la figure 3.21. Les temps de propagation susceptibles d'être atteints par ces PAL sont du même ordre de grandeur que ceux vus figure 3.18 pour la famille 16R8.

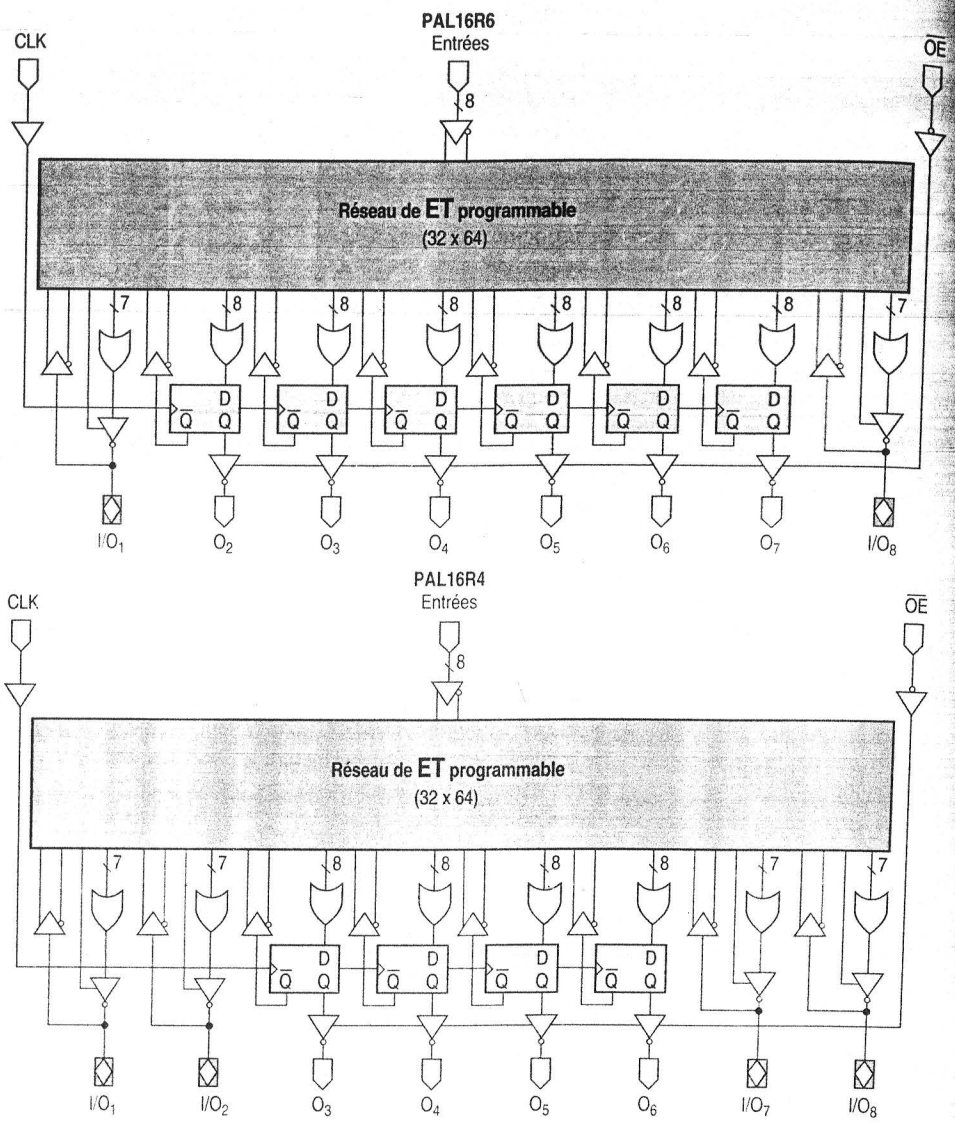


Figure 3.14 :
Organisation interne des 16R4 et 16R6.

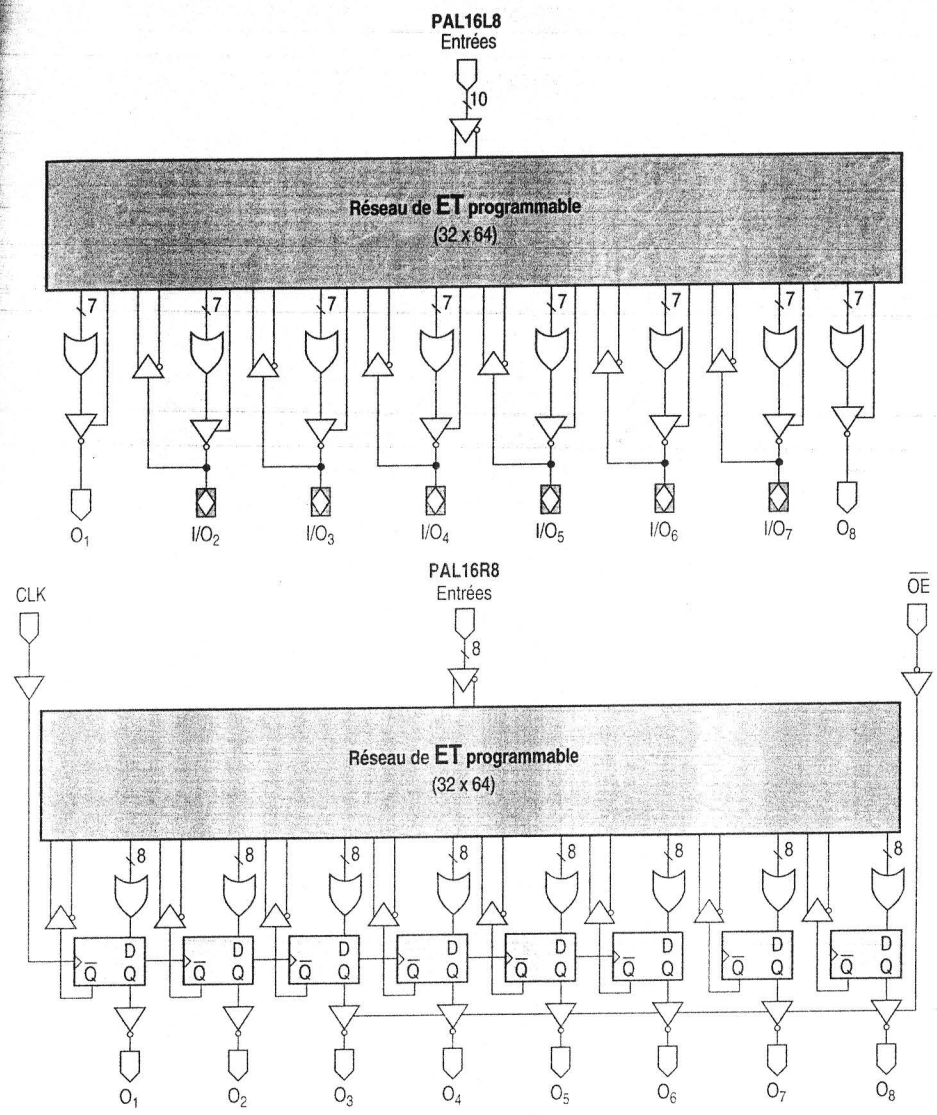
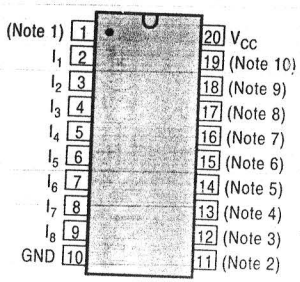
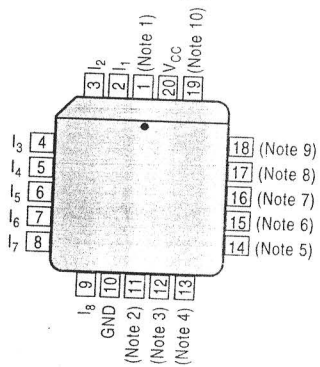


Figure 3.15 :
Organisation interne des 16L8 et 16R8.

DIP



20 -Pin PLCC



Note	16L8	16R8	16R6	16R4
1	I_0	CLK	CLK	CLK
2	I_9	$\overline{OE}$	$\overline{OE}$	$\overline{OE}$
3	O_1	O_1	I/O_1	I/O_1
4	I/O_2	O_2	O_2	I/O_2
5	I/O_3	O_3	O_3	O_3
6	I/O_4	O_4	O_4	O_4
7	I/O_5	O_5	O_5	O_5
8	I/O_6	O_6	O_6	O_6
9	I/O_7	O_7	O_7	I/O_7
10	O_8	O_8	I/O_8	I/O_8

28-Pin PLCC

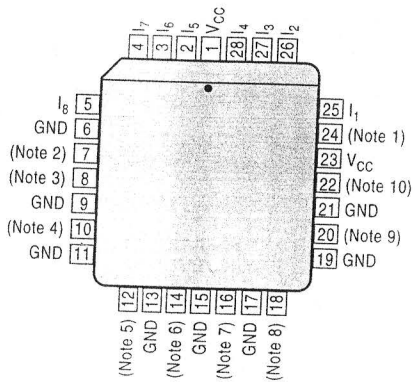


Figure 3.16 :
Brochages de la famille 16R8.

16R8 (-4)

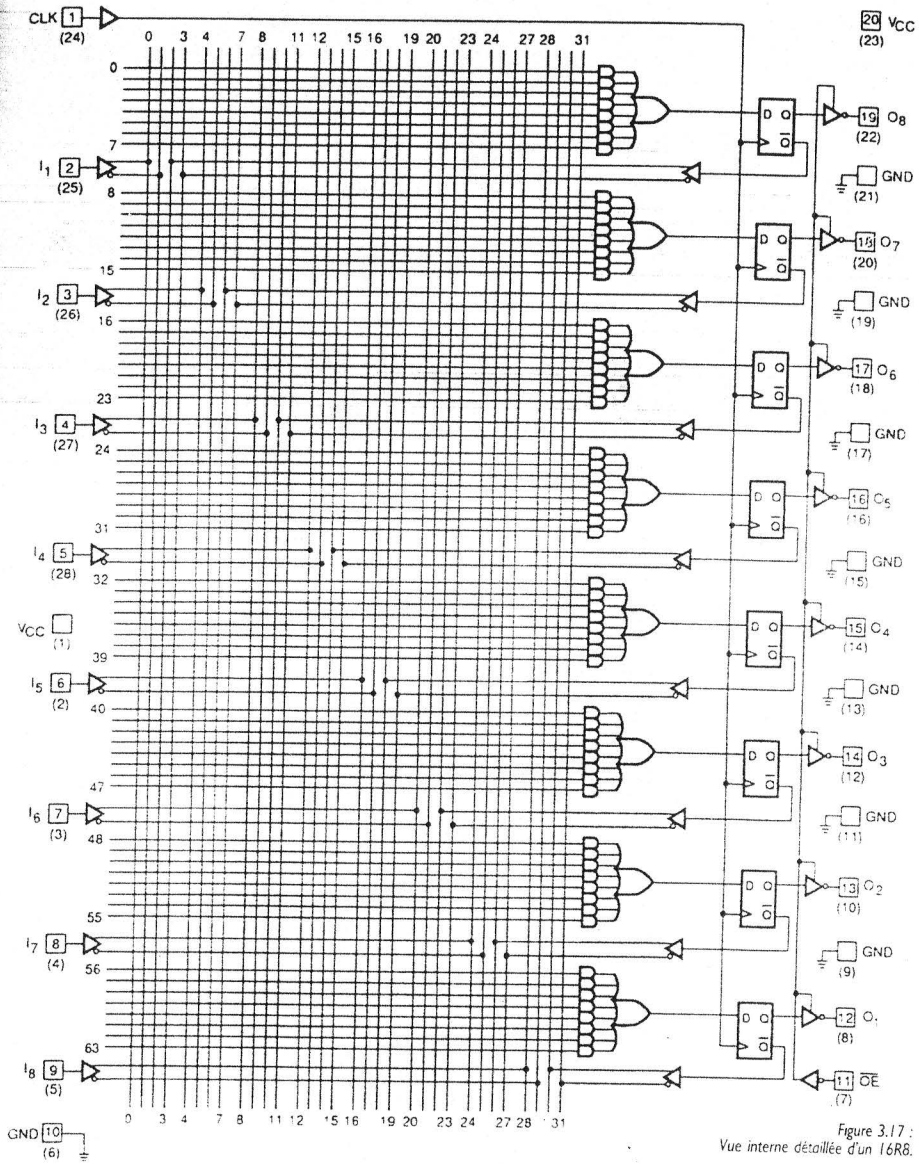


Figure 3.17 :
Vue interne détaillée d'un 16R8.

Parameter Symbol	Parameter Description			-5 Min. (Note 3)	-5 Max.	-4 Min. (Note 3)	-4 Max.	Unit
tPD	Input or Feedback to Combinatorial Output		16L8, 16R8, 16R4	1	5	1	4.5	ns
ts	Setup Time from Input or Feedback to Clock			4.5		4.5		ns
tH	Hold Time			0		0		ns
tCO	Clock to Output			1	4.0	1	3.5	ns
tSKEWR	Skew Between Registered Outputs (Note 4)				1		0.5	ns
tWL	Clock Width	LOW	16R8, 16R6, 16R4	4		4		ns
tWH		HIGH		4		4		ns
tMAX	Maximum Frequency (Note 5)	External Feedback 1/(ts + tco)		117		125		MHz
		Internal Feedback (fCNT)		125		125		MHz
		No Feedback 1/(twH + twL)		125		125		MHz
tPZX	$\overline{OE}$ to Output Enable			1	6.5	1	6.5	ns
tPXZ	$\overline{OE}$ to Output Disable			1	5	1	5	ns
tEA	Input to Output Enable Using Product Term Control		16L8, 16R6, 16R4	2	6.5	2	6.5	ns
tER	Input to Output Disable Using Product Term Control			2	5	2	5	ns

Figure 3.18 :
Paramètres de commutation des PAL 16R8.

Référence	Entrées dédiées	Sorties	Termes produits par sortie	Retour	Validation des sorties
20L8	14	6 combinatoires 2 combinatoires	7 7	E/S –	Programmable Programmable
20R8	12	8 à registres	8	Registres	Patte
20R6	12	6 à registres 2 combinatoires	8 7	Registres E/S	Patte Programmable
20R4	12	4 à registres 4 combinatoires	8 7	Registres E/S	Patte Programmable

Figure 3.19 :
Les membres de la famille des PAL 24 pattes 20R8.

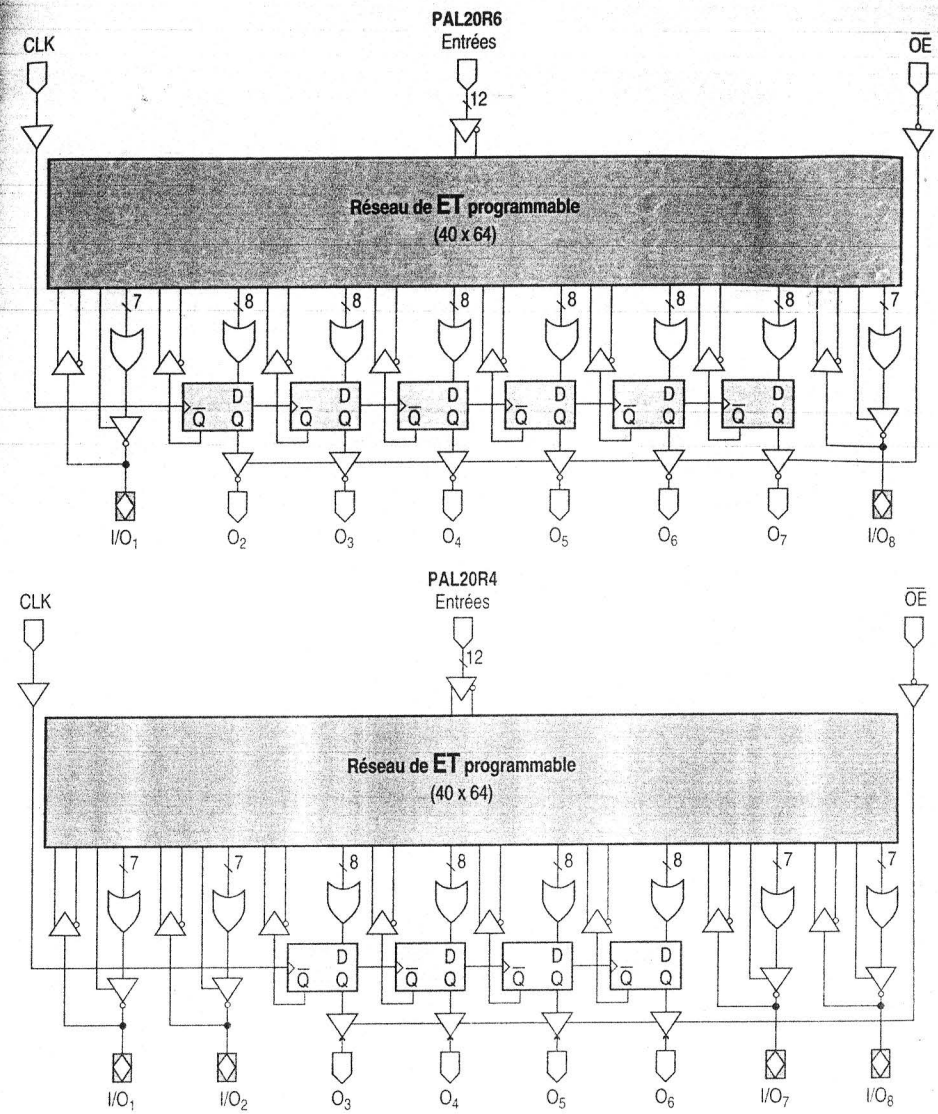


Figure 3.20 :
Organisation interne des 20R4 et 20R6.

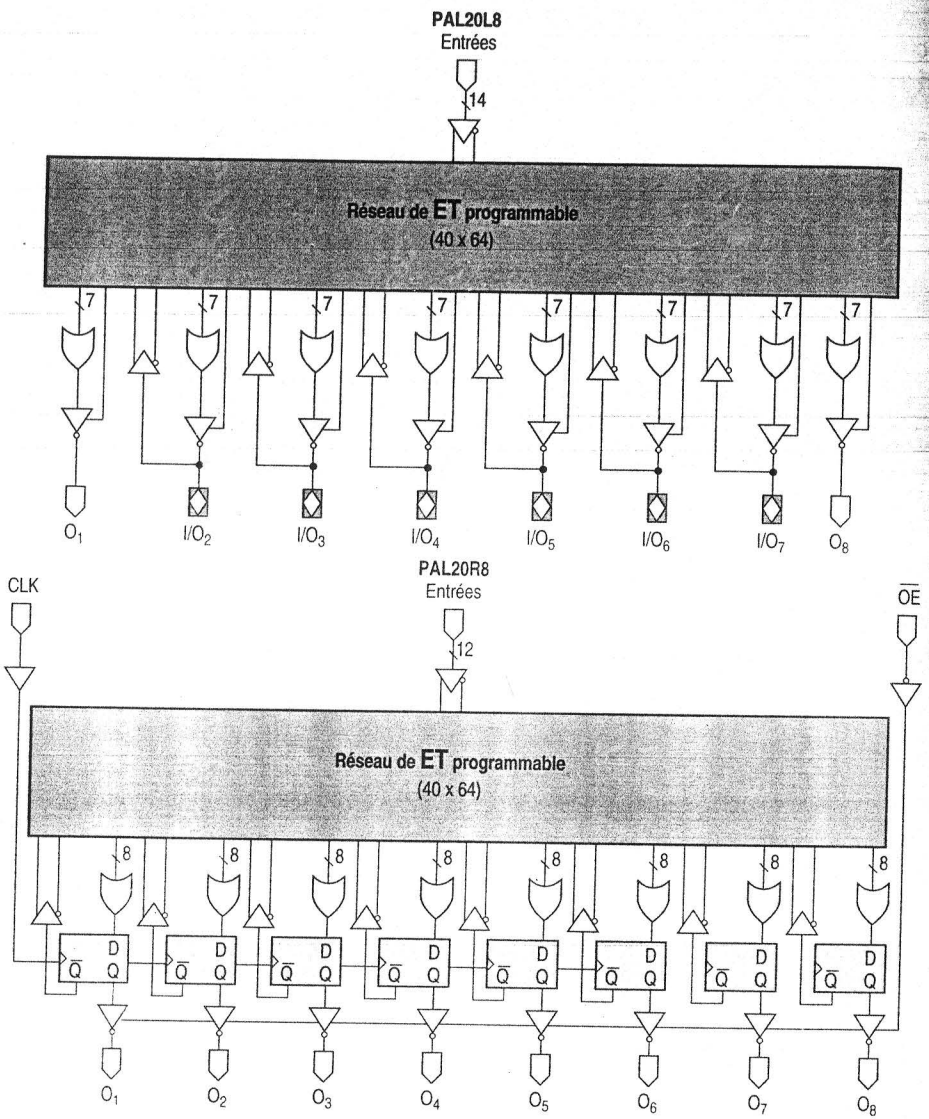
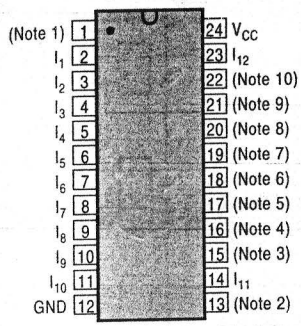


Figure 3.21 :
Organisation interne des 20L8 et 20R8.

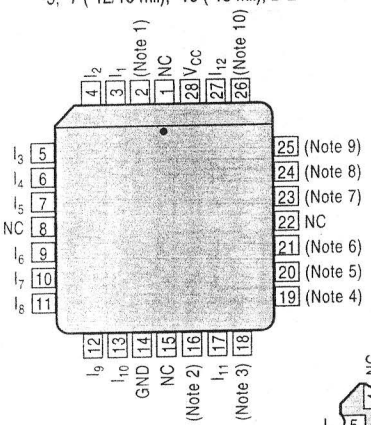
SKINNYDIP/FLATPACK



Note	20L8	20R8	20R6	20R4
1	I_0	CLK	CLK	CLK
2	I_{13}	$\overline{OE}$	$\overline{OE}$	$\overline{OE}$
3	O_1	O_1	I/O_1	I/O_1
4	I/O_2	O_2	O_2	I/O_2
5	I/O_3	O_3	O_3	O_3
6	I/O_4	O_4	O_4	O_4
7	I/O_5	O_5	O_5	O_5
8	I/O_6	O_6	O_6	O_6
9	I/O_7	O_7	O_7	I/O_7
10	O_8	O_8	I/O_8	I/O_8

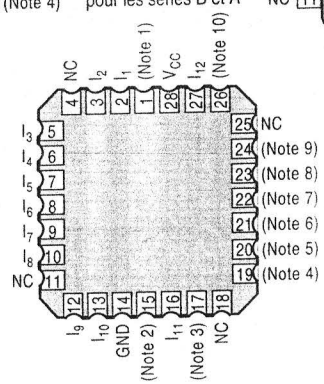
PLCC/LCC

JEDEC : valable seulement pour les séries -5, -7 (-12/10 mil), -10 (-15 mil), B-2



LCC

Valable seulement pour les séries B et A



PLCC

Valable seulement pour les séries B et A

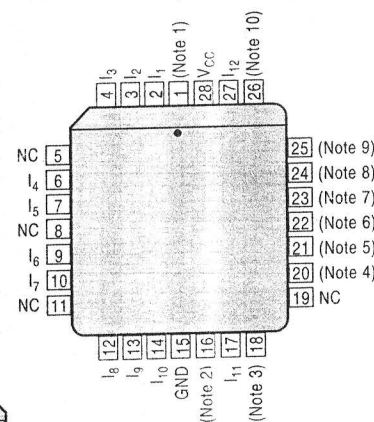


Figure 3.22 :
Brochages de la famille 20R8.

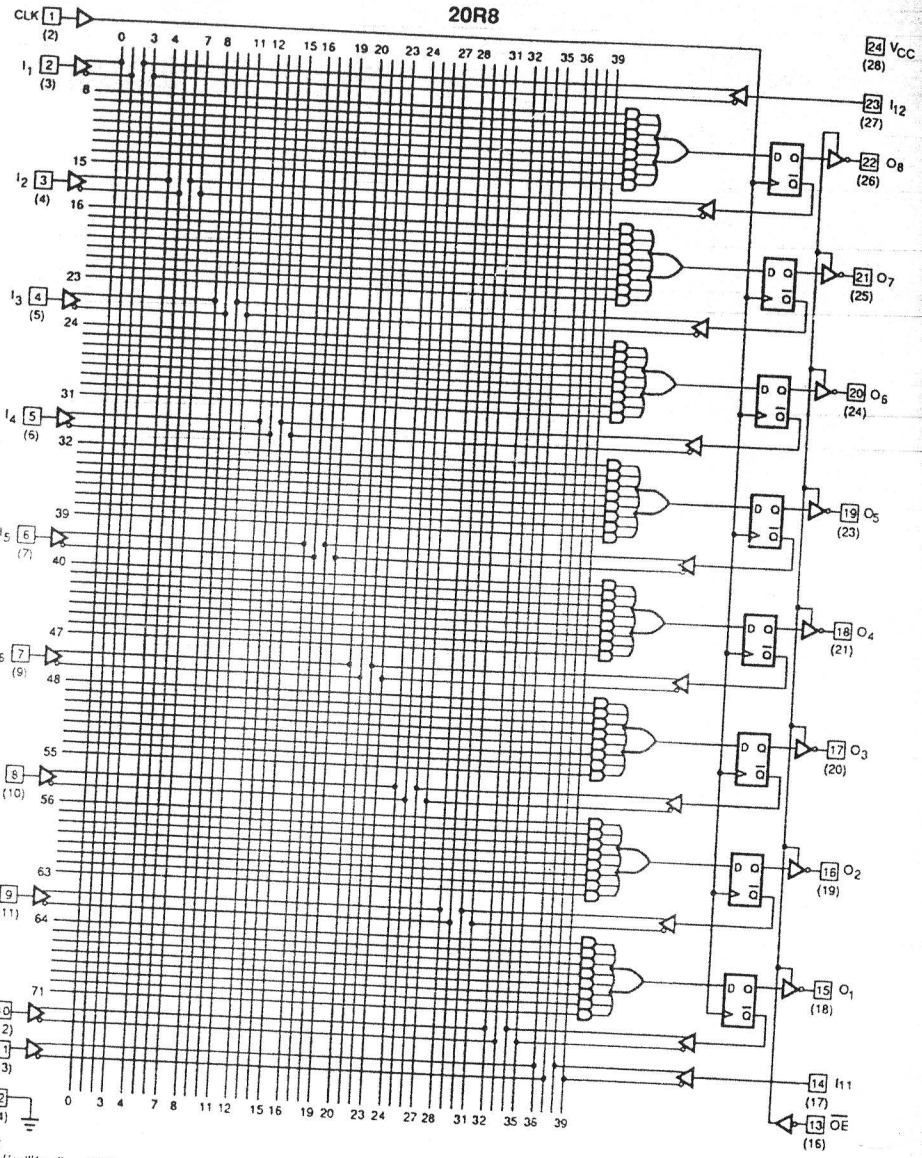


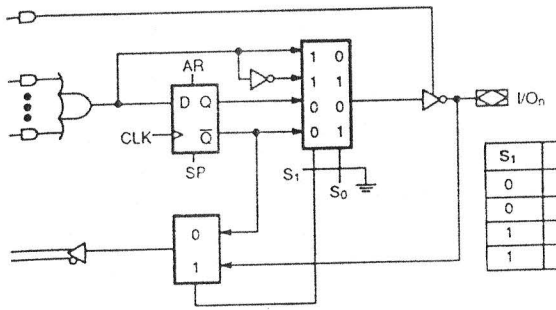
Figure 3.23 :
Vue interne détaillée d'un 20R8.

Nous ne nous étendrons pas plus longtemps sur ces types de PAL dont les différentes architectures de sorties ont été présentées au paragraphe précédent. En outre, elles ne posent vraiment aucun problème particulier puisque ce ne sont que des portes et/ou des bascules analogues à celles utilisées en logique classique.

LES PAL VERSATILES OU VPAL

Ces PAL, dont le plus célèbre est très certainement le 22V10 d'AMD qui a été le premier circuit de ce type introduit sur le marché, présentent une évolution des PAL vers les circuits logiques programmables de plus haut niveau. En effet, ils continuent à respecter les principes généraux présentés tout au long de ce chapitre mais utilisent une structure de cellule de sortie qui s'apparente de plus en plus aux macro-cellules que nous découvrirons dans la suite de cet ouvrage dans d'autres circuits logiques programmables.

La figure 3.24 va nous permettre de voir quelles sont les nombreuses possibilités de configuration de cette cellule de sortie.



S_1	S_0	Configuration des sorties
0	0	Registre, actif bas
0	1	Registre, actif haut
1	0	Combinatoire, actif bas
1	1	Combinatoire, actif haut

Figure 3.24 :
Structure de la cellule de sortie d'un PAL versatile ou VPAL.

Nous y voyons bien évidemment une bascule ou flip-flop pré-positionnable associé à deux multiplexeurs programmables. Le multiplexeur à quatre entrées permet de configurer la cellule selon un des quatre modes présentés sur cette même figure. On programme pour cela les *signaux* S_0 et S_1 qui sont en fait générés grâce à des fusibles internes, exactement comme d'autres liaisons du PAL.

La structure de la cellule prend alors l'un des aspects présentés sur la figure 3.25 sur laquelle nous reconnaissons toutes les architectures de sorties des PAL classiques déjà vus.

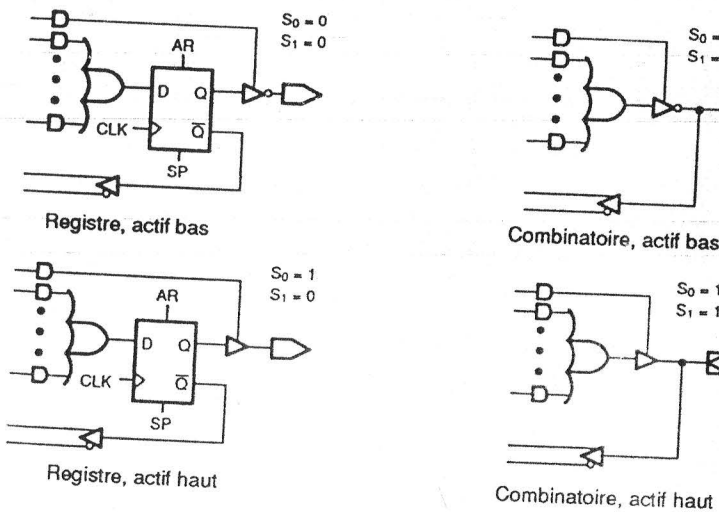


Figure 3.25 :
Les différentes configurations permises par un PAL versatile ou VPAL

Les VPAL disposent d'une possibilité de pré-positionnement à un ou à zéro de la bascule de sortie ; possibilité qui est toutefois commune à toutes les bascules d'un même circuit. Afin de disposer d'un maximum de souplesse d'utilisation, ces commandes de pré-positionnement ne sont pas disponibles sur une patte spécifique du boîtier, mais sont ramenées sur la zone de fusibles.

Afin de permettre aux VPAL de démarrer dans un état bien défini, une circuiterie de pré-positionnement automatique à la mise sous tension est en outre incluse dans le boîtier. Lors d'une mise sous tension, toutes les sorties Q des bascules sont donc automatiquement placées au niveau bas.

Ceci étant vu, nous vous présentons figure 3.26 la structure interne du PAL 22V10 qui est le seul circuit de ce type actuellement disponible sur le marché en technologie bipolaire. Son succès est tel qu'il est produit par de nombreux fabricants dont Texas Instruments et AMD entre autres.

Compte tenu de l'intérêt de ce PAL et de sa versatilité (qui lui a d'ailleurs valu son appellation et son sigle distinctif), vous êtes en droit de vous demander pourquoi il n'en existe qu'une seule version. La réponse à cette question est double :

- D'une part, compte tenu de sa structure actuelle, le 22V10 peut remplacer la majorité des autres PAL bipolaires toutes familles confondues.
- D'autre part, ce circuit est apparu sur le marché à peu près en même temps que les premiers GAL et PAL CMOS ; sa structure a donc été largement utilisée dans les PAL CMOS où elle était relativement plus facile à reproduire en grand nombre que dans les PAL bipolaires. De ce fait, on rencontre d'autres versions de PAL V, mais en technologie PAL CMOS que vous découvrirez au chapitre suivant.

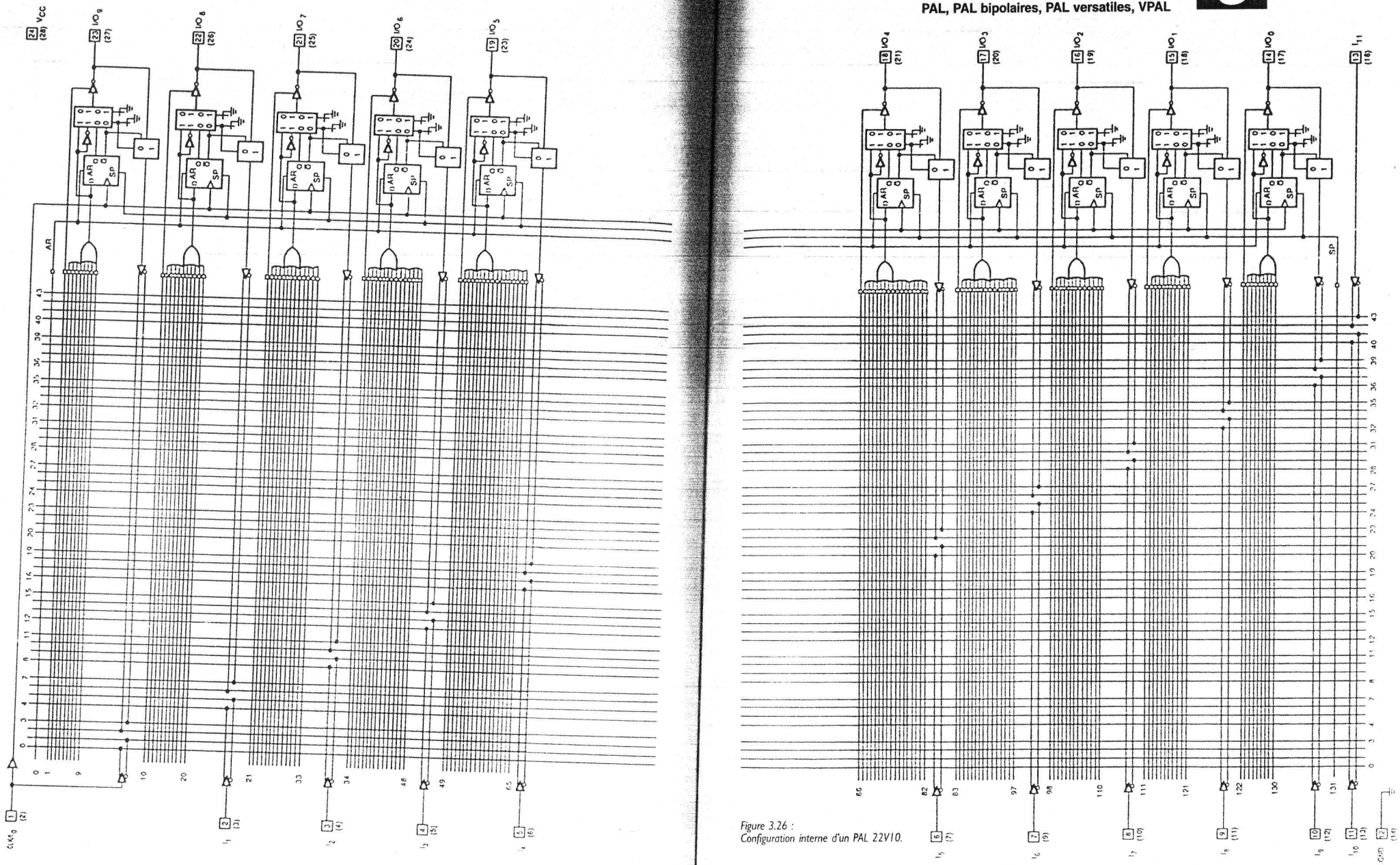


Figure 3.26 :
Configuration interne d'un PAL 22V10.

UN EXEMPLE D'APPLICATION

Compte tenu de la densité d'intégration permise par les PAL les plus récents que sont les PAL 24 pattes ou bien encore les VPAL, il est vite très difficile, dans le cadre d'un ouvrage comme celui-ci, de présenter des exemples d'applications complets. En effet, de nombreuses pages sont alors nécessaires pour présenter les choix retenus et les listings des équations logiques, et cela ne sert finalement pas à grand chose, sauf si vous avez la chance d'avoir besoin exactement de la même application.

Nous avons donc décidé de ne retenir qu'un exemple simple qui reste facilement publiable et qui n'est là que pour vous montrer tout à la fois la puissance des PAL, même de petite taille, et leur facilité de mise en œuvre. Si vous souhaitez d'autres exemples, sachez que tous le fabricants de PAL éditent des notes d'applications, voire parfois des recueils entiers regroupant des exemples d'applications de provenance très diverse.

Décodeur BCD hexadécimal pour afficheurs 7 segments

Cette application vise à réaliser un circuit qui reçoit en entrée un code BCD et génère les signaux nécessaires pour commander des afficheurs 7 segments classiques. Le décodage n'est pas seulement décimal mais hexadécimal et permet donc l'affichage des lettres A à F. Le circuit doit en outre disposer de trois entrées de contrôle :

- Une entrée lampe-test permettant l'allumage de tous les segments à des fins de contrôle.
- Une entrée de contrôle d'intensité de l'affichage permettant de mettre celui-ci en, ou hors fonction, compte tenu du niveau logique appliqué. Il est ainsi possible de faire facilement du multiplexage ou de la commande de luminosité par variation de largeur d'impulsions.
- Une entrée RBI qui permet, par connexion à la sortie RBO du circuit précédent, de réaliser l'effacement des zéros non significatifs.

De ce fait, notre circuit doit également disposer d'une sortie RBO afin de permettre de cascader les circuits pour cette suppression de zéro.

Le tableau de la figure 3.27 donne la table de décodage à réaliser au niveau des segments des afficheurs, tandis que la figure 3.28 montre

D_3	D_2	D_1	D_0	Chiffre	Segments
0	0	0	0	0	ABCDEF
0	0	0	1	1	BC
0	0	1	0	2	ABDEG
0	0	1	1	3	ABCDG
0	1	0	0	4	BCFG
0	1	0	1	5	ACDFG
0	1	1	0	6	ACDEFG
0	1	1	1	7	ABC
1	0	0	0	8	ABCDEFG
1	0	0	1	9	ABCDFG
1	0	1	0	A	ABCEFG
1	0	1	1	B	CDEFG
1	1	0	0	C	ADEF
1	1	0	1	D	BCDEG
1	1	1	0	E	ADEFG
1	1	1	1	F	AEFG

Figure 3.27 : Table de décodage BCD-hexadécimale vers afficheurs 7 segments.

un exemple-type d'utilisation de plusieurs circuits de ce type interconnectés.

La réalisation de ce circuit ne présente aucune difficulté majeure et peut être menée à bien avec un simple 16L8 comme le montre le listing de conception présenté figure 3.29. Nous reconnaissons sans trop de difficulté sur celui-ci les zones de définition du circuit, d'affectation des pattes puis les équations logiques. Après traitement par PALASM, le tableau de programmation est généré.

Pour information nous avons reproduit en figure 3.30 le schéma interne du PAL une fois programmé. Même s'il est encore possible d'essayer de le suivre, vous vous rendez compte que nous atteignons la limite de ce qu'il est possible de faire *à la main*. Le recours à un logiciel de conception et de programmation de PAL est donc pleinement justifié dès que l'application devient un tant soit peu complexe. Et encore, ne sommes-nous ici qu'en logique combinatoire qui reste donc simple à analyser.

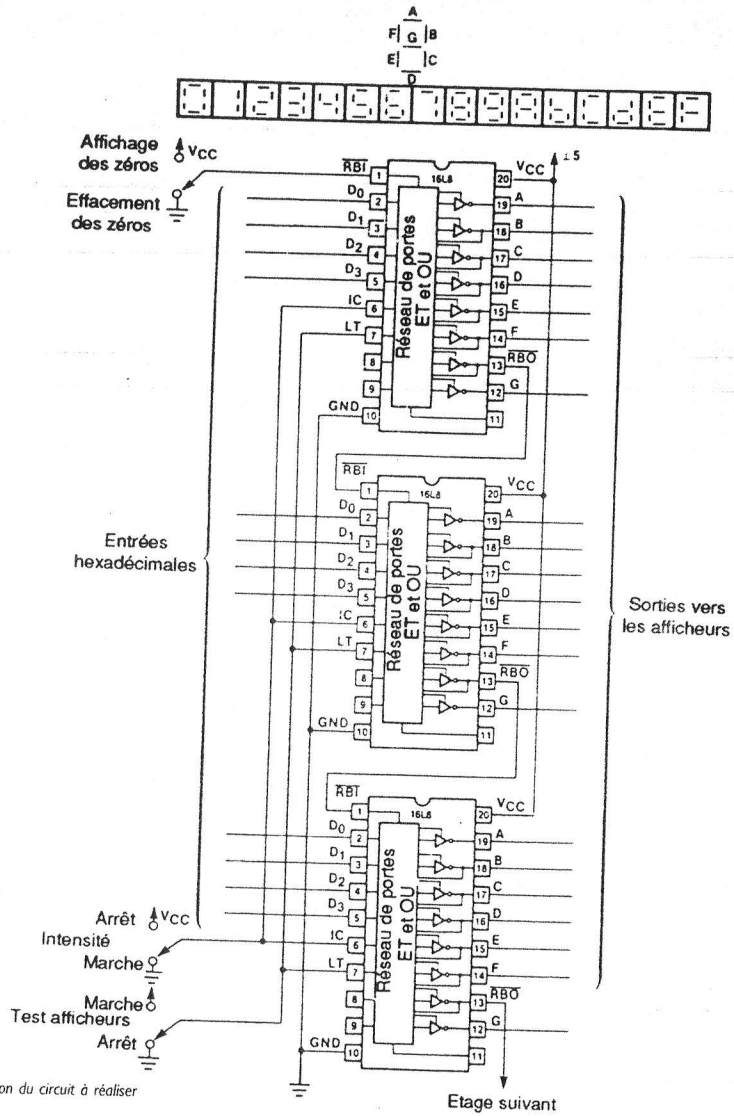


Figure 3.28 :
Schéma d'utilisation du circuit à réaliser (doc. NS).


```
PALASM VERSION 1.5

PAL16L8
PAT07
HEX
BLANK
/RBI DO D1 D2 D3 IC LT NC NC GND
NC G /RBO F E D C B A VCC
IF(/IC)/A=/RBO*/DO*/D2+/RBO*/DO*D3+/RBO*D1*D2+
        /RBO*D1*D2*/D3+/RBO*DO*D2*/D3+/RBO*/D1*/D2*D3+LT
IF(IC)/B=/RBO*/D2*/D3+/RBO*/DO*/D2+/RBO*/DO*/D1*/D3+
        /RBO*DO*D1*/D3+/RBO*DO*/D1*/D3+LT
IF(IC)/C=/RBO*DO*/D1+/RBO*DO*/D2+/RBO*/D1*/D2+
        /RBO*D2*/D3+/RBO*/D2*D3+LT
IF(IC)/D=/RBO*/D1*D3+/RBO*/DO*/D2*/D3+
        /RBO*DO*D1*/D2+/RBO*/DO*D1*D2+/RBO*DO*/D1*D2+LT
IF(IC)/E=/RBO*/DO*/D2+/RBO*D2*D3+/RBO*/DO*D1+
        /RBO*D1*D3+LT
IF(IC)/F=/RBO*/DO*/D1+/RBO*/D2*D3+/RBO*D1*D3+
        /RBO*/DO*D2+/RBO*/D1*D2*/D3+LT
IF(VCC)RBO=/DO*/D1*/D2*/D3*/RBI
IF(/IC)/G=/RBO*D1*/D2+/RBO*DO*D3+/RBO*/D2*D3+
        /RBO*/DO*D1+/RBO*/D1*D2*/D3+LT
DESCRIPTION
```

Figure 3.29 :
Listing de définition du décodeur (doc. NS).

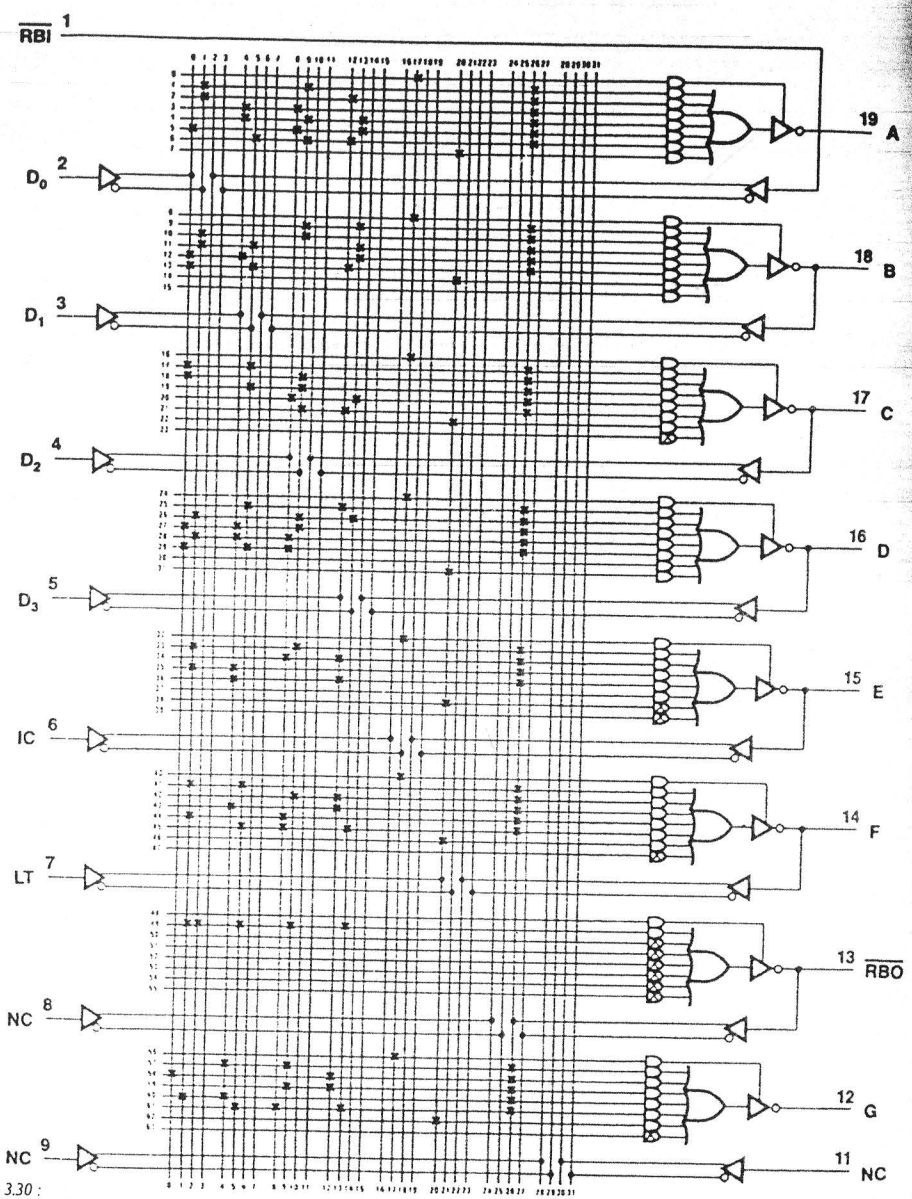


Figure 3.30 :
Programmation du 16L8 en décodeur (doc. NS).

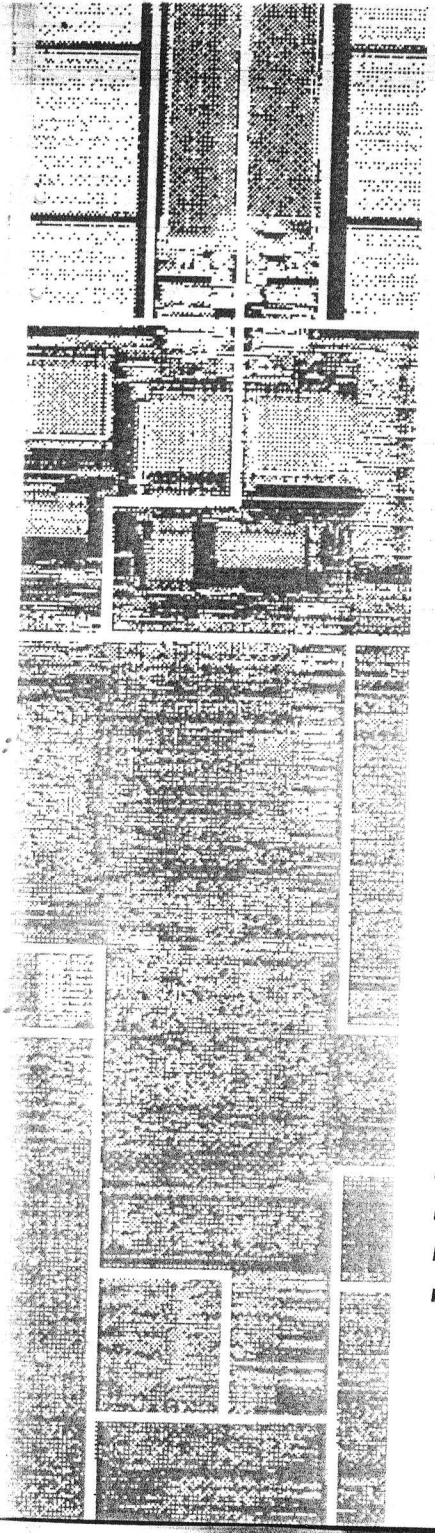
CIRCUITS LOGIQUES PROGRAMMABLES

CHAPITRE 4

Les GAL et PAL CMOS.

EEPAL, E2PAL, GAL, PAL CMOS, PAL EECMOS, PAL E2CMOS, ispGAL.

- Notions de base 118
- Les GAL ou PAL CMOS classiques 119
- Le GAL ou PAL CMOS asynchrone à registre 20RA10 . 127
- L'ispGAL 22V10 130
- Les GAL 6001 et 6002 ou GAL à architecture de type FPLA 134



Alors que les circuits dont nous avons parlé jusqu'à présent existent depuis quinze ans, voire parfois vingt ans pour certains d'entre eux, les GAL et les PAL CMOS objets de ce chapitre, sont d'apparition beaucoup plus récente puisqu'ils ont été mis sur le marché il y a seulement dix ans.

En effet, ils ont été commercialisés pour la première fois en 1985 par la société américaine Lattice Semiconductor dont le terme GAL est d'ailleurs une marque déposée. Ce terme en lui-même ne signifie pas grand chose par rapport à ce que nous avons pu vous expliquer jusqu'à présent, puisque c'est l'abréviation de Generic Array Logic, ce que l'on peut traduire mot-à-mot par réseau logique générique.

En fait, si vous avez bien lu ce qui précède, vous avez déjà certainement compris que ces GAL ne sont rien d'autre que des PAL CMOS, c'est-à-dire des PAL programmables, mais surtout effaçables électriquement. Ils partagent avec eux les mêmes références et, même si des détails technologiques font qu'ils ne sont pas réalisés tout à fait de la même façon et si les vrais GAL Lattice sont parfois un peu plus performants que les PAL CMOS d'autres fabricants, ils sont fonctionnellement équivalents, à références identiques bien sûr.

Les GAL et PAL CMOS.

EEPAL, E2PAL, GAL, PAL CMOS, PAL EECMOS, PAL E2CMOS, ispGAL.

Ceci étant, la société Lattice dispose tout de même de GAL particuliers n'ayant aucun équivalent en PAL CMOS. Nous traiterons bien évidemment ces circuits à la fin du présent chapitre. Par ailleurs, cette société propose depuis 1994 environ, des circuits appelés pLSI et ispLSI pour in system programmable LSI, c'est-à-dire circuits logiques à grande échelle programmables dans le système où ils sont installés. Ces circuits sont à mi-chemin entre les EPLD et les FPGA, mais étant tout de même plus proches des EPLD, ils seront donc traités avec ces derniers au chapitre 5.

Vous trouverez tout de même dans ce chapitre un circuit isp mais pour la simple raison que c'est un ispGAL. Il est donc logique de l'étudier avec les GAL.

NOTIONS DE BASE

Les GAL ont été introduits sur le marché à une époque où les PAL CMOS étaient encore quasiment inconnus. Le but visé par Lattice avec cette technologie était quadruple :

- Offrir des produits fabriqués selon une technologie autorisant des vitesses de travail comparables à celles des PAL bipolaires tout en étant testables à 100 %, y compris pour la zone de programmation, lors de la fabrication. Ceci est évidemment impossible avec les PAL bipolaires comme nous l'avons vu, puisque cela équivaudrait à faire sauter les fusibles en phase de fabrication !
- Permettre un remplacement, au moins fonctionnel, mais idéalement broche pour broche, des PAL bipolaires dans n'importe quelle application.
- Offrir une consommation beaucoup plus faible que les PAL bipolaires de complexité équivalente.
- Proposer une plus grande souplesse de configuration des entrées/sorties que les PAL bipolaires.

La société Lattice est parvenue à respecter ces quatre critères, puisque les GAL sont maintenant repris sous forme de PAL CMOS par quasiment tous les autres fabricants de PAL bipolaires.

À la date de rédaction de cet ouvrage, la gamme des GAL comporte neuf familles de produits présentées dans le tableau de la figure 4.1. Six membres de ces familles sont en tous points comparables aux PAL CMOS de mêmes références. Il serait d'ailleurs plus logique de dire que ce sont les PAL CMOS qui sont comparables à ces GAL, étant donné l'ordre relatif de développement des produits.

Les trois familles de produits restantes, quant à elles, sont pour l'instant typiques des GAL et nous leur réserverons donc une place à part dans ce chapitre.

Comme les PAL bipolaires, les GAL et PAL CMOS peuvent être développés avec divers logiciels qui peuvent être soit spécifiques, soit plus généraux. Ainsi, les logiciels de CAO classiques tels que Orcad, Pcad, Futurenet, etc. supportent-ils sans problème le concept de GAL ou de PAL CMOS, tant en phase de conception qu'en phase de test.

Toujours comme les PAL bipolaires et pour éviter les copies dont nous avons parlé au chapitre 3, les GAL et PAL CMOS disposent eux

Référence	Nb de pattes	Tpd (ns)	Fmax (MHz)	I_{cc} (mA)	I_{sb} (µA)	Description
GAL16V8	20	5, 7,5, 10, 15, 25	166, 100, 62,5, 41,6	55, 90, 115	–	PLD universel 20 pattes
GAL16V8Z/ZD	20	12, 15	83,3 62,5	55	100	PLD 20 pattes faible consommation
GAL16VP8	20	15, 25	80, 50	115	–	16V8 avec sorties 64 mA
GAL18V10	20	15, 20	62,5	115	–	Sous-ensemble du 22V10
GAL20RA10	24	10, 15, 20, 30	71,4 50, 41,7, 25	100	–	Asynchrone à registre
GAL20V8	24	5, 7,5, 10, 15, 25	166, 100, 62,5, 41,6	55, 90, 115	–	PLD universel 24 pattes
GAL20V8Z/ZD	24	12, 15	83,3 62,5	55	100	PLD 24 pattes faible consommation
GAL20VP8	24	15, 25	80, 50	115	–	20V8 avec sorties 64 mA
GAL20XV10	24	10, 15, 20	100, 83,3, 71,4	90	–	24 pattes à OU Exclusif
GAL22V10	24	5, 6, 7,5, 10, 15, 25	200, 166, 142,8, 105, 83,3, 38,5	55, 90, 130, 140, 150	–	PLD universel 24 pattes
ispGAL22V10	28	7,5, 10, 15	111, 105, 83,3	140	–	22V10 programmable en circuit
GAL26CV12	28	7,5, 10, 15, 20	142.8 105 83,3, 62,5	130	–	Sur-ensemble du 22V10
GAL6001	24	30	27	150	–	Architecture FPLA
GAL6002	24	15, 20	75, 60	135	–	Architecture FPLA

Figure 4.1 :
Les différentes familles de GAL proposées par Lattice.

aussi d'un bit de protection ou de sécurité qui, une fois programmé, interdit toute lecture du contenu du circuit par quelque méthode que ce soit. Ce bit ne peut évidemment jamais être effacé seul. Il n'est donc effacé que lors d'un effacement global du GAL, ce qui assure bien sa protection contre toute duplication illicite.

LES GAL OU PAL CMOS CLASSIQUES

Ces circuits référencés pour la partie *racine* : 16V8, 20V8, 18V10, 22V10 et 26CV12 existent dans la famille des GAL et des PAL CMOS. Ils préfigurent les circuits logiques plus évolués que nous allons découvrir dans la suite de cet ouvrage avec l'apparition de la notion de macro-cellule, c'est-à-dire en fait de bloc logique configurable et à priori polyvalent.

L'architecture générale est encore comparable, par contre, à celle des PAL, comme le montre la figure 4.2 qui présente celle des 16V8 et

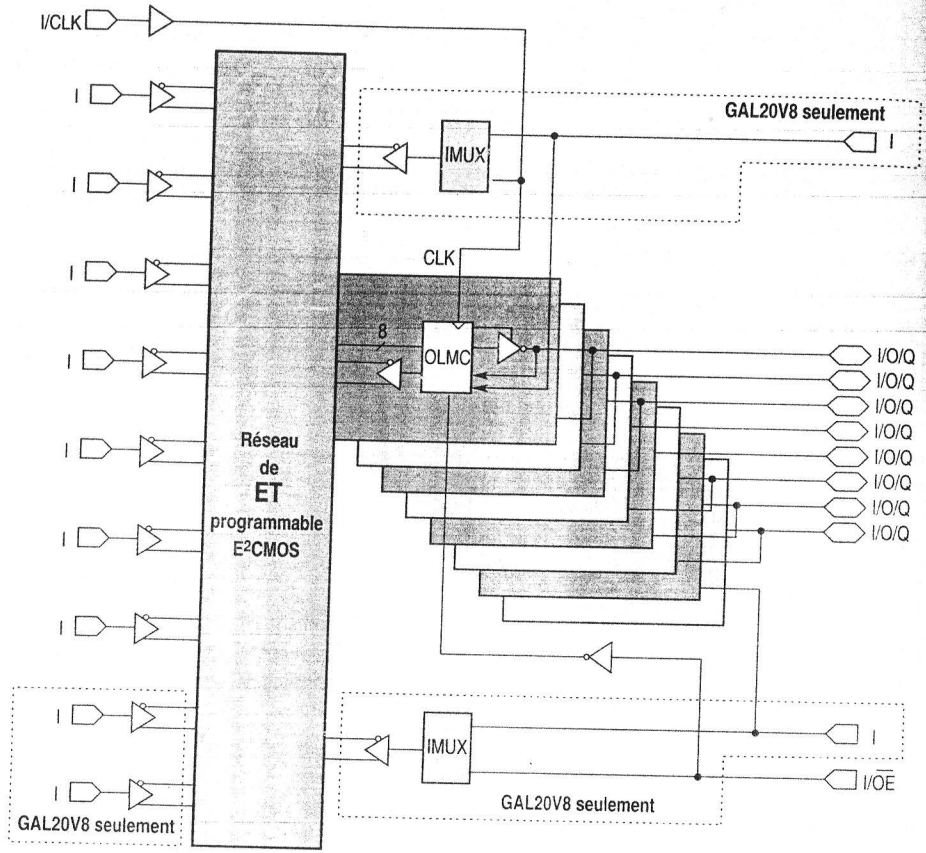


Figure 4.2 :
Architecture interne des GAL et PAL CMOS 16V8 et 20V8.

20V8. On y voit, en effet, une zone d'interconnexions programmables de type ET, en technologie E2CMOS, qui est le pendant de la zone de fusibles des PAL. Cette zone permet de relier entre elles des macro-cellules appelées OLMC par Lattice pour *Output Logic MacroCell*.

Cette macro-cellule dont le schéma vous est présenté figure 4.3 est composée d'une bascule, de portes OU, d'un buffer de sortie, et surtout de multiples multiplexeurs ou commutateurs programmables permettant de lui faire prendre de nombreuses configurations différentes.

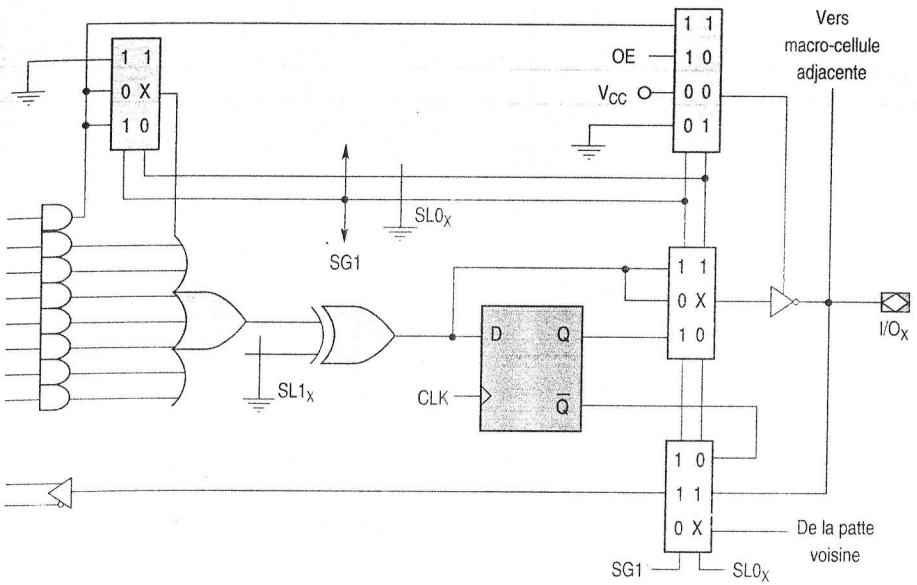


Figure 4.3 :
Cellule de sortie des GAL et PAL CMOS de type V.

Ces configurations vous sont présentées figure 4.4. Il est à noter toutefois que toutes les macro-cellules doivent être configurées de la même façon au sein d'un même circuit. Il n'est ainsi pas possible d'avoir par exemple trois sorties de type à registre et cinq sorties de type combinatoire. Cette relative limitation peut être résolue par l'utilisation des PAL CMOS ou GAL 18V10, 22V10 ou 26CV12 dont nous parlerons dans un instant.

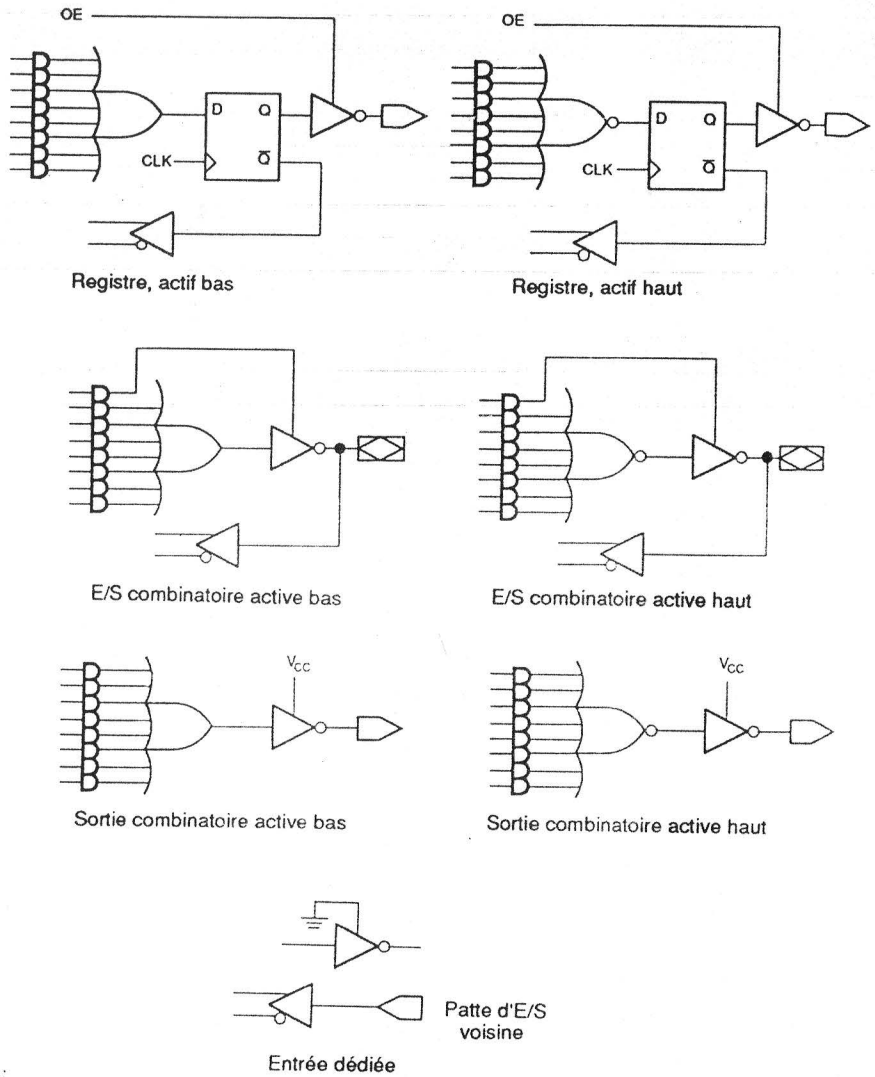


Figure 4.4 :
Les différentes configurations de la cellule de sortie ou OLMC d'un GAL de type V.

Malgré cela, l'examen détaillé de ces configurations conjoint à celui de l'architecture interne des GAL vous permettrait de constater qu'en fait, nos GAL ou PAL CMOS 16V8 et 20V8 sont capables de remplacer tous les PAL bipolaires classiques des familles 16R8 et 20R8 ainsi que d'anciennes références aujourd'hui disparues, soit un total de 42 références de PAL bipolaires différentes.

L'intérêt des GAL ou PAL CMOS, outre leur possibilité d'effacement, apparaît donc ici aussi au niveau de la gestion des stocks et des approvisionnements qui se trouvent considérablement simplifiés.

La mise en œuvre des GAL ou PAL CMOS ne dépayse pas l'utilisateur déjà habitué aux PAL bipolaires. Le brochage d'un 16V8 est ainsi identique à celui d'un PAL bipolaire 20 pattes classiques comme vous pouvez le constater figure 4.5. Quant à ce qui est du schéma interne présenté figure 4.6, toujours dans le cas du 16V8 pris en exemple, on croirait celui d'un banal 16R8 au remplacement près de la structure fixe de sortie par les macro-cellules ou OLMC.

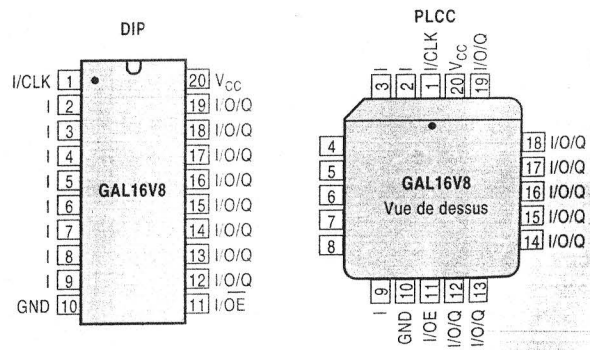


Figure 4.5 :
Brochages des GAL 16V8.

Les performances que l'on peut attendre de ces GAL ou PAL CMOS sont loin d'être ridicules en termes de vitesse puisque, comme le montre la figure 4.7, elles rivalisent sans difficulté avec celles des PAL bipolaires (comparez cette figure avec la figure 3.18, par exemple).

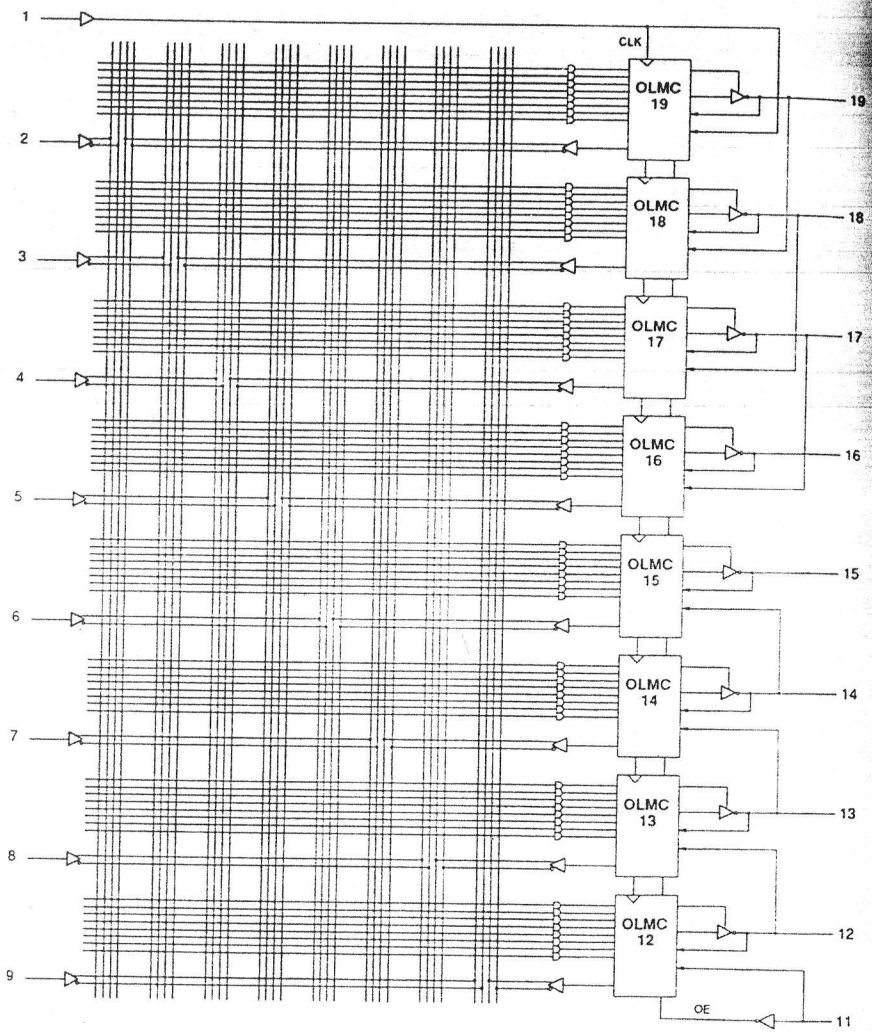


Figure 4.6 :
Schéma interne d'un GAL 16V8.

Paramètres	Test	Description	COM -5 Min	COM -5 Max	IND -7 Min	IND -7 Max	Unités
t_{pd}	A	Input or I/O to combinational Output	1	5	1	7,5	ns
t_{co}	A	Clock to Output Delay	1	4	1	5	ns
t_{cf}[2]	–	Clock to Feedback Delay	–	3	–	3	ns
t_{su}	–	Setup Time, Input or Feedback before Clock ↑	3	–	7	–	ns
t_h	–	Hold Time, Input or Feedback after Clock ↑	0	–	0	–	ns
f_{max}[3]	A	Maximum Clock Frequency with External Feedbacck, 1/(tsu + tco)	142,8	–	83,3	–	MHz
	A	Maximum Clock Frequency with Internal Feedbacck, 1/(tsu + tcf)	166	–	100	–	MHz
	A	Maximum Clock Frequency with No Feedbacck	166	–	100	–	MHz
t_{wh}	–	Clock Pulse Duration, High	3	–	5	–	ns
t_{wl}	–	Clock Pulse Duration, Low	3	–	5	–	ns
t_{en}	B	Input or I/O to Output Enabled	1	6	1	9	ns
	B	$\overline{OE}$ to Output Enabled	1	6	1	6	ns
t_{dis}	C	Input or I/O to Output Disabled	1	5	1	9	ns
	C	$\overline{OE}$ to Output Disabled	1	5	1	6	ns

Figure 4.7 :
Principales caractéristiques de commutation des GAL 16V8.

Les PAL CMOS ou GAL 18V10, 22V10 et 26CV12 adoptent, quant à eux, une architecture légèrement différente, visible figure 4.8, ainsi qu'un schéma de macro-cellule un peu plus élaboré afin d'être identique à celle du célèbre PAL bipolaire versatile 22V10.

Le GAL ou PAL CMOS 22V10 est donc fonctionnellement identique au PAL bipolaire 22V10 tandis que le 18V10 en est une version 20 pattes, suffisante dans bien des cas et plus économique, alors que le 26CV12 est une version 28 pattes, offrant ainsi plus de lignes d'entrées/sorties.

Les possibilités de configuration de cette macro-cellule sont rappelées figure 4.9. Outre la souplesse permise, notez aussi qu'une possibilité de positionnement à un ou à zéro de chaque bascule de sortie est offerte et que les lignes correspondantes sont ramenées sur la zone d'interconnexions programmables.

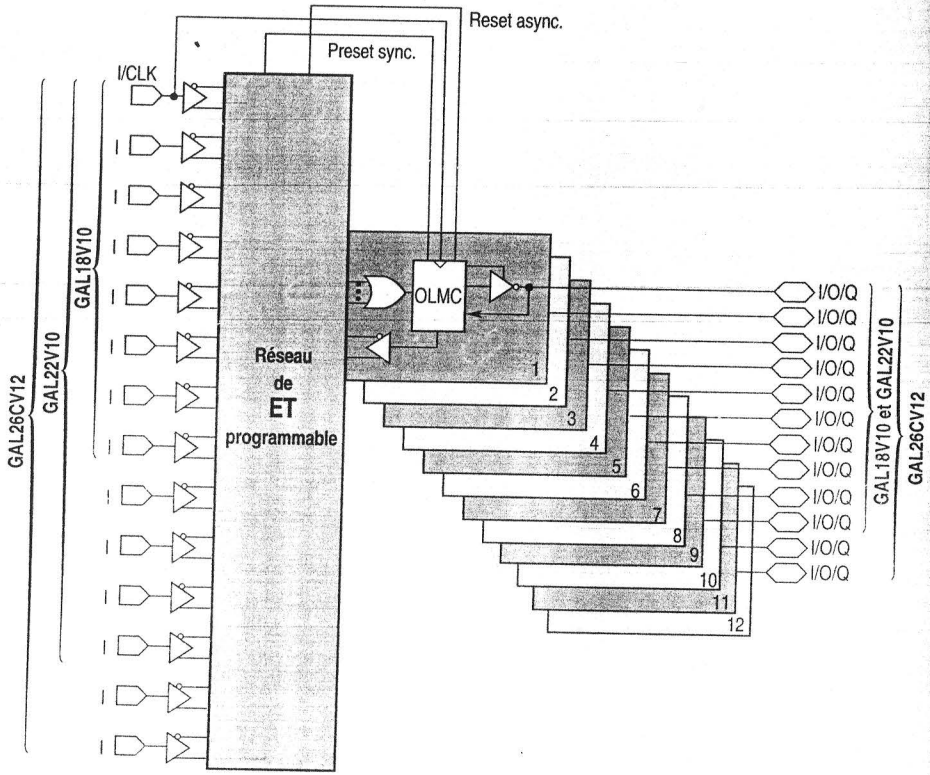


Figure 4.8 :
Architecture interne des GAL 18V10, 22V10 et 26CV12.

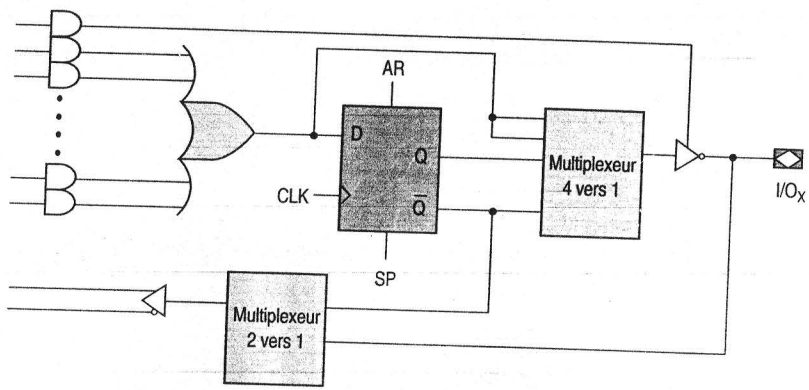


Figure 4.9 :
Schéma de la macro-cellule d'un GAL ou PAL CMOS 22V10.

LE GAL OU PAL CMOS ASYNCHRONE À REGISTRE 20RA10

Ce circuit, qui respecte bien évidemment les concepts généraux des GAL ainsi que nous venons de le voir, utilise une macro-cellule différente de celle des GAL de la série V ; en effet, comme le laisse clairement supposer sa référence, c'est un GAL destiné à remplacer les PAL bipolaires asynchrones à registres qui n'existent plus aujourd'hui.

Il fait donc appel à une structure de cellule de sortie présentée figure 4.10 que nous allons analyser rapidement. Nous y reconnaissons bien sûr une structure de PAL à registre, ce qui est logique vu l'appellation retenue pour cette famille de produits. Mais, par rapport aux PAL à registres classiques, des possibilités supplémentaires sont offertes.

Tout d'abord le registre dispose de deux lignes de pré-positionnement : AR pour le reset ou positionnement à zéro, et AP pour le preset ou positionnement à un. Ces lignes proviennent de la matrice de fusibles et peuvent donc être commandées par les signaux ou combinaisons de signaux de votre choix.

Outre ces fonctions *normales*, ces deux lignes AR et AP jouent un autre rôle. En effet, si elles sont toutes les deux mises à un, le flip-flop est invalidé, ou transparent si vous préférez, et le PAL se transforme alors en modèle combinatoire simple.

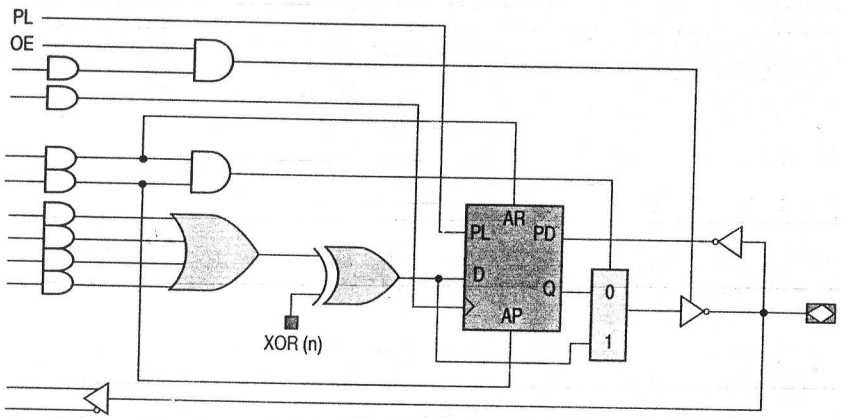


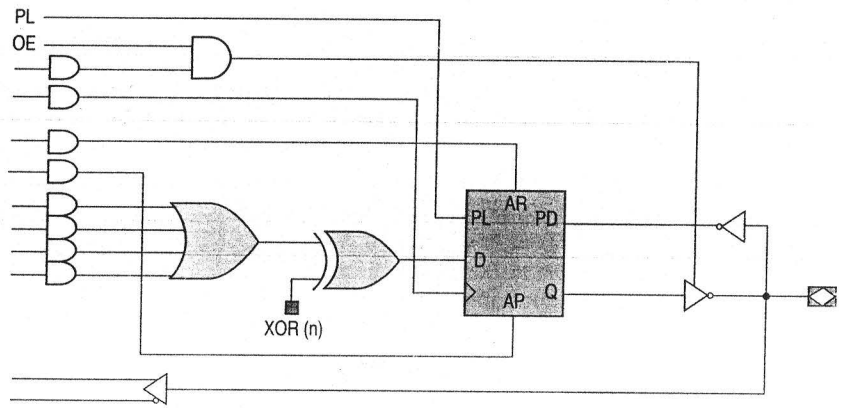
Figure 4.10 :
Structure de la cellule de sortie d'un PAL asynchrone à registre.

L'horloge provient elle aussi de la zone de fusibles et peut donc être la combinaison de n'importe quels signaux.

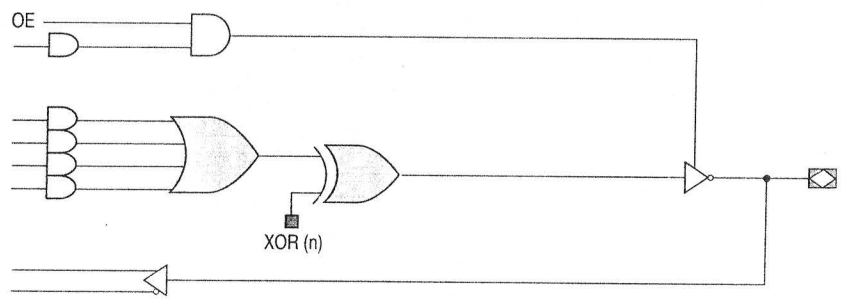
La sortie est évidemment de type trois états, mais sous contrôle de deux signaux de provenance différente. Le premier signal appelé OE est un signal global commun à toutes les sorties du PAL. Il ne provient pas de la zone de fusibles, mais correspond à une patte particulière du boîtier. Le second signal provient par contre de la zone de fusibles et peut donc servir de mise en trois états locale pour la sortie considérée. Le signal OE global est évidemment prioritaire étant donné son mode de câblage.

Compte tenu de tout cela, il est possible de configurer n'importe quelle sortie d'une PAL asynchrone à registre comme un PAL à registre ou comme un PAL combinatoire comme le montre la figure 4.11.

Outre le fait qu'il permette de réaliser des fonctions qui ne pouvaient pas l'être avec les GAL ou PAL CMOS vus précédemment, le GAL asynchrone à registre peut également remplacer dans la majorité des cas n'importe quel PAL à registre ou combinatoire, à nombre d'entrées/sorties identiques bien entendu.



Configuration de la macro-cellule de sortie (à registre avec polarité)



Configuration de la macro-cellule de sortie (combinatoire avec polarité)

Figure 4.11 :
Les différentes configurations permises par un PAL asynchrone à registre.

L'ISPGAL 22V10

Comme sa référence le laisse supposer, l'ispGAL 22V10 est en fait un GAL 22V10 programmable en circuit en raison de la présence du préfixe isp. À la date de rédaction de cet ouvrage, ce circuit est proposé seulement par Lattice, mais il est probable qu'à plus ou moins longue échéance, d'autres fabricants lui emboîteront le pas, comme ce fut le cas avec les PAL CMOS et les GAL.

L'ispGAL 22V10 est strictement identique au PAL CMOS ou GAL 22V10 pour ce qui est de l'architecture interne et de la configuration des macro-cellules logiques, et nous ne présenterons donc pas celles-ci à nouveau dans ce paragraphe. Comme le montre la figure 4.12, le brochage est également compatible de celui du 22V10 classique mais seulement pour ce qui est du boîtier LCC sur lequel les quatre lignes nécessaires à la programmation occupent des pattes laissées libres sur le 22V10 classique. Le boîtier DIL du 22V10, quant à lui, étant *plein*, il n'est pas proposé pour l'isp22V10.

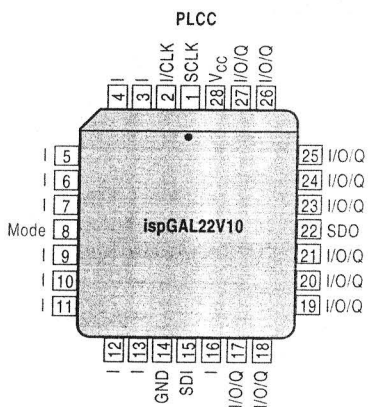


Figure 4.12 :
Brochage d'un ispGAL 22V10.

Le principe de programmation en circuit adopté par Lattice fait appel à quatre lignes de contrôle : MODE, SDI, SDO et SCLK. Le rôle exact de ces lignes ne présente que peu d'intérêt pour l'utilisateur car, comme nous allons le voir, le logiciel nécessaire à leur commande est quasiment toujours fourni prêt à l'emploi.

Ces lignes doivent en effet être contrôlées correctement afin de permettre la programmation et l'effacement du circuit. Pour ce faire, deux solutions principales sont utilisables.

La première, schématisée figure 4.13, est envisageable dans le cas de l'utilisation du GAL dans une application comportant au moins un circuit *intelligent*, microprocesseur ou microcontrôleur par exemple. Ce dernier exécute alors le logiciel approprié pour piloter les quatre lignes du GAL et le programmer conformément aux besoins de l'application.

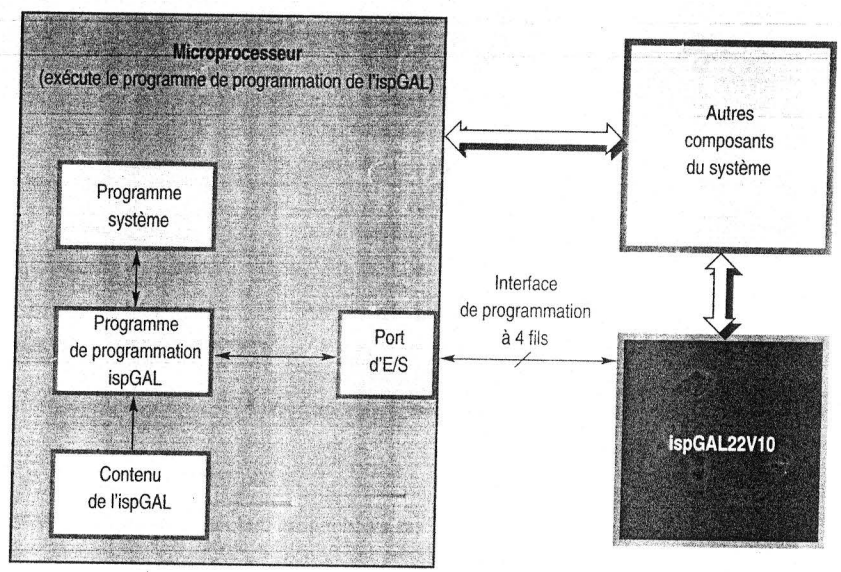


Figure 4.13 :
Principe de programmation d'un ispGAL à partir du système dans lequel il est intégré.

Afin que l'utilisateur ait un minimum de travail de programmation à réaliser, Lattice fournit gratuitement les différents modules logiciels nécessaires, écrits en langage C ANSI standard, qui sont ainsi portables et compilables par la majorité des outils de développement pour microprocesseurs et microcontrôleurs.

La deuxième solution, schématisée figure 4.14, consiste à faire appel à un micro-ordinateur compatible PC connecté aux quatre lignes de programmation via son port parallèle imprimante grâce à une interface fort simple. Le schéma de cette interface est présenté en figure 4.15 pour vous en convaincre !

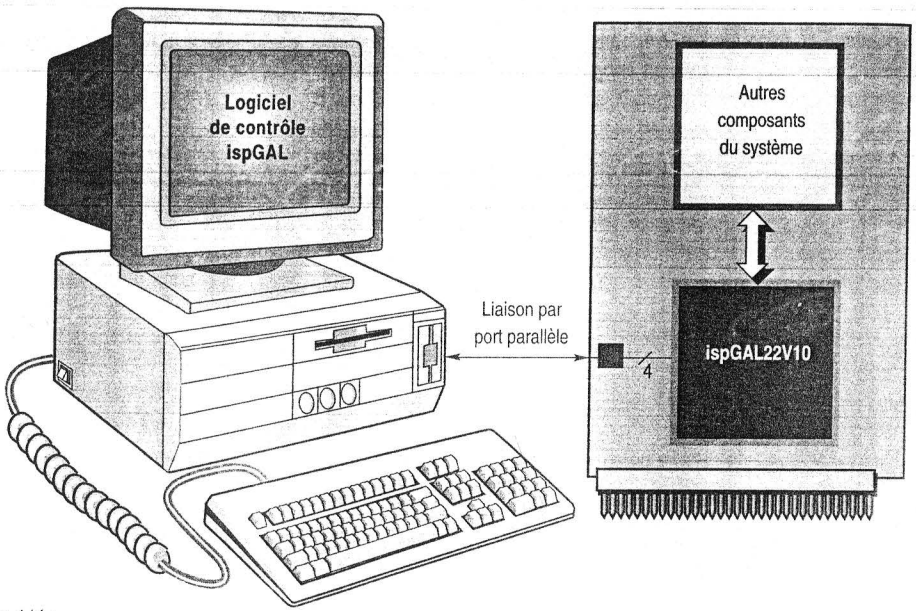


Figure 4.14 :
Principe de programmation d'un ispGAL à partir d'un compatible PC.

Dans ce cas, il est évidemment toujours possible d'utiliser les modules de programmes en C que nous venons d'évoquer, mais Lattice fournit également le logiciel nécessaire prêt à l'emploi.

La programmation de plusieurs ispGAL placés dans un même système est une opération relativement fréquente, aussi sachez qu'il est possible de la réaliser très simplement par une connexion *à la chaîne* de ces circuits comme il est montré figure 4.16. Il ne faut ainsi, toujours, que quatre fils pour programmer nos ispGAL, quel que soit leur nombre.

Figure 4.15 :
Schéma de l'interface de programmation d'un ispGAL à partir du port parallèle d'un compatible PC.

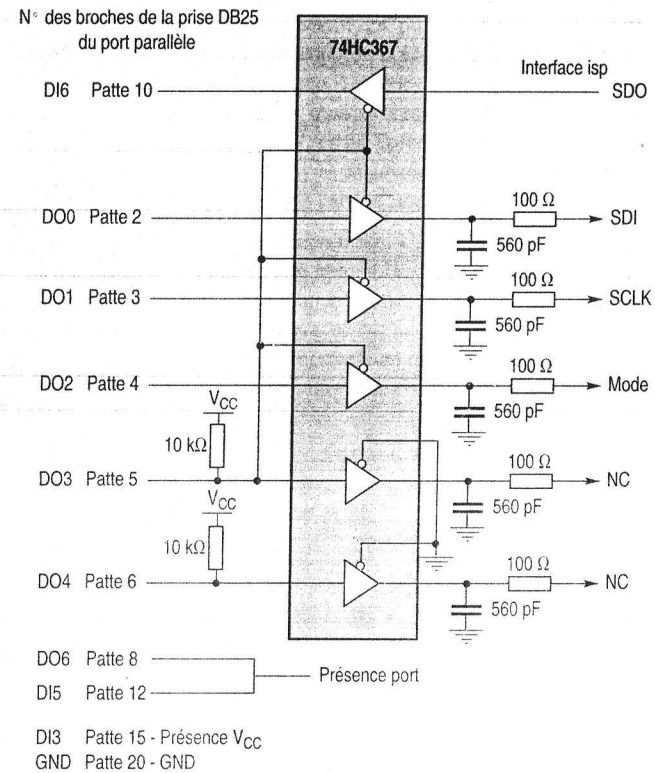
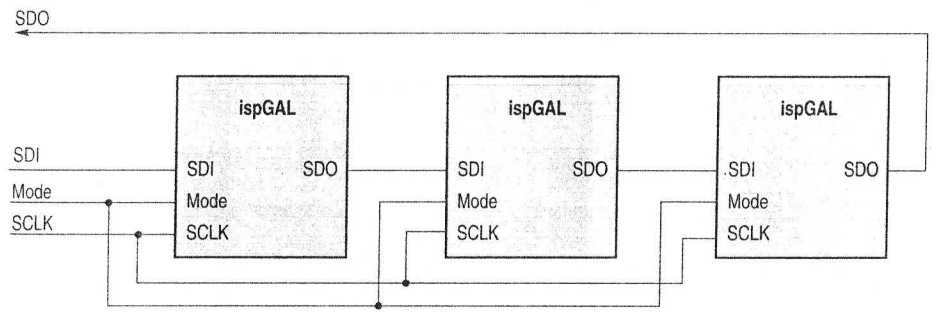


Figure 4.16 :
Principe de programmation en cascade de plusieurs ispGAL.



LES GAL 6001 ET 6002 OU GAL À ARCHITECTURE DE TYPE FPLA

Au début du chapitre précédent nous vous avons parlé de la structure baptisée FPLA dans laquelle le réseau de ET d'entrées était programmable au même titre que le réseau de OU de sorties.

Le GAL 6001 de Lattice est un exemple de circuit utilisant une structure de type FPLA, beaucoup plus souple d'emploi, bien sûr, que les structures de PAL ou GAL classiques, et particulièrement performante dans le cadre de machines d'états ou de moteurs d'automates.

Voyons quelles sont les ressources disponibles dans un GAL 6001 :

- 10 macro-cellules de sorties appelées OLMC.
- 8 macro-cellules enterrées (nous verrons par la suite la signification de ce terme et le rôle de ces cellules) appelées BLMC.
- 20 macro-cellules d'entrées et d'entrées/sorties appelées respectivement ILMC et IOLMC.
- 2 entrées d'horloges indépendantes, une pour les cellules d'entrées appelée ICLK, une pour les cellules de sorties et enterrées appelée OCLK.

La figure 4.17 présente la structure détaillée de ce circuit. Son apparente complexité est due au fait que, comme il s'agit d'un circuit de type FPLA, les OU de sorties et les ET d'entrées sont programmables. Ce sont eux qui occupent la majeure partie de cette figure.

La figure 4.18, quant à elle, présente la structure-type d'une macro-cellule d'entrée ou d'entrée/sortie ainsi que les trois configurations qu'elle peut adopter, par programmation bien sûr. On constate que les entrées peuvent être de type simple comme sur un PAL classique par exemple. Dans ce cas, le signal appliqué à la patte concernée se retrouve injecté directement, sous forme vraie et inversée, bien sûr, à la zone programmable.

Il existe également un mode entrée à registre dans lequel le signal est appliqué à l'entrée d'une bascule D. La commande d'horloge de cette bascule est commune à toute les macro-cellules d'entrées et se trouve ramenée sur la patte spécifique du boîtier ICLK. Le signal appliqué à l'entrée se retrouve donc envoyé sur la zone programmable après un front montant du signal d'horloge.

Le dernier mode est le mode latch qui, pour ce qui est de la représentation, ne diffère pas beaucoup du précédent. En fait, seul change le fonctionnement de la bascule. Lorsque le signal d'horloge est au niveau bas, cette dernière est transparente et se laisse traverser par le signal appliqué sur la patte d'entrée. Lorsque l'horloge est au niveau haut, la bascule est verrouillée et sa sortie conserve l'état antérieur même si le niveau appliqué à la patte d'entrée change.

Les macro-cellules de sortie et les macro-cellules enterrées utilisent le même schéma de base présenté figure 4.19, conjointement aux diverses configurations possibles. Si la notion de macro-cellule de sortie ne vous est pas inconnue, il n'en est pas de même de celle de macro-cellule enterrée. En fait, ces cellules reçoivent en entrées les signaux issus de la zone de OU programmables de sortie du circuit et leurs sorties alimentent les entrées de la zone de ET programmables d'entrée.

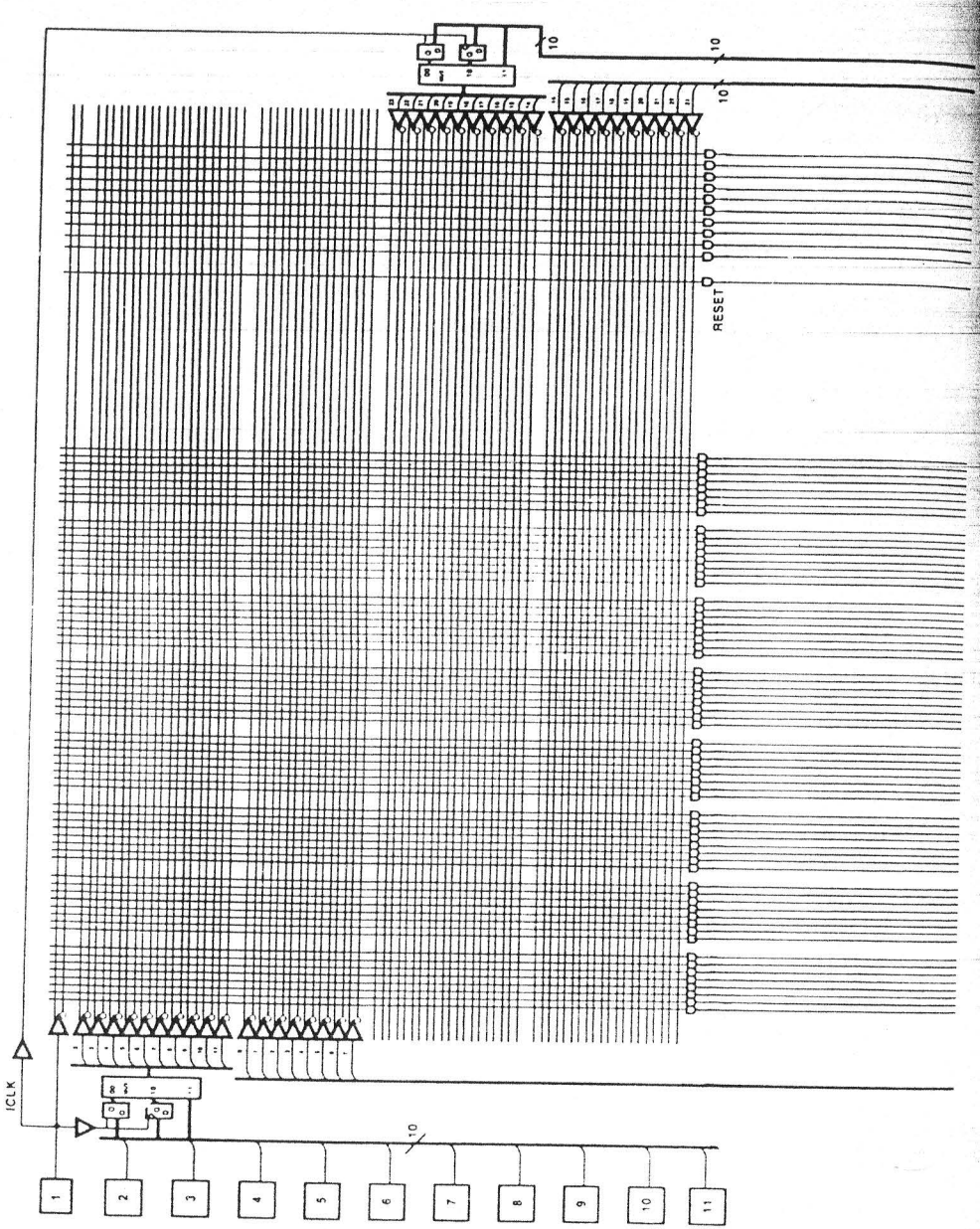
Pour préciser exactement leur emplacement, examinez à nouveau la figure 4.17 sur laquelle nous avons encerclé une de ces macro-cellules. Remarquez que, vu leur emplacement, elles permettent de réaliser des rebouclages internes extrêmement souples et performants.

Ces cellules peuvent fonctionner dans trois modes différents. Le mode combinatoire qui est le plus simple et dans lequel toutes les bascules et registres sont invalidés. Ce mode dispose d'une entrée OU exclusif permettant de programmer la polarité du signal, mais uniquement sur les macro-cellules de sortie.

Le deuxième mode est un mode à registre de type D/E avec une commande d'horloge commune qui provient de la patte spécifique OCLK du boîtier. Cette commande d'horloge est commune à toutes les macro-cellules de sorties et enterrées, mais elle est distincte de celle des macro-cellules d'entrées.

Le troisième mode, enfin, est encore un mode à registre, mais cette fois-ci, avec une commande d'horloge qui provient directement de la zone programmable. Dans les deux cas, et comme pour le mode combinatoire, une fonction OU Exclusif est prévue pour permettre une sélection de polarité du signal, mais ici encore, uniquement sur les macro-cellules de sortie.

Ceci étant vu, les zones programmables méritent elles aussi un peu d'attention en raison de leurs tailles assez inhabituelles.



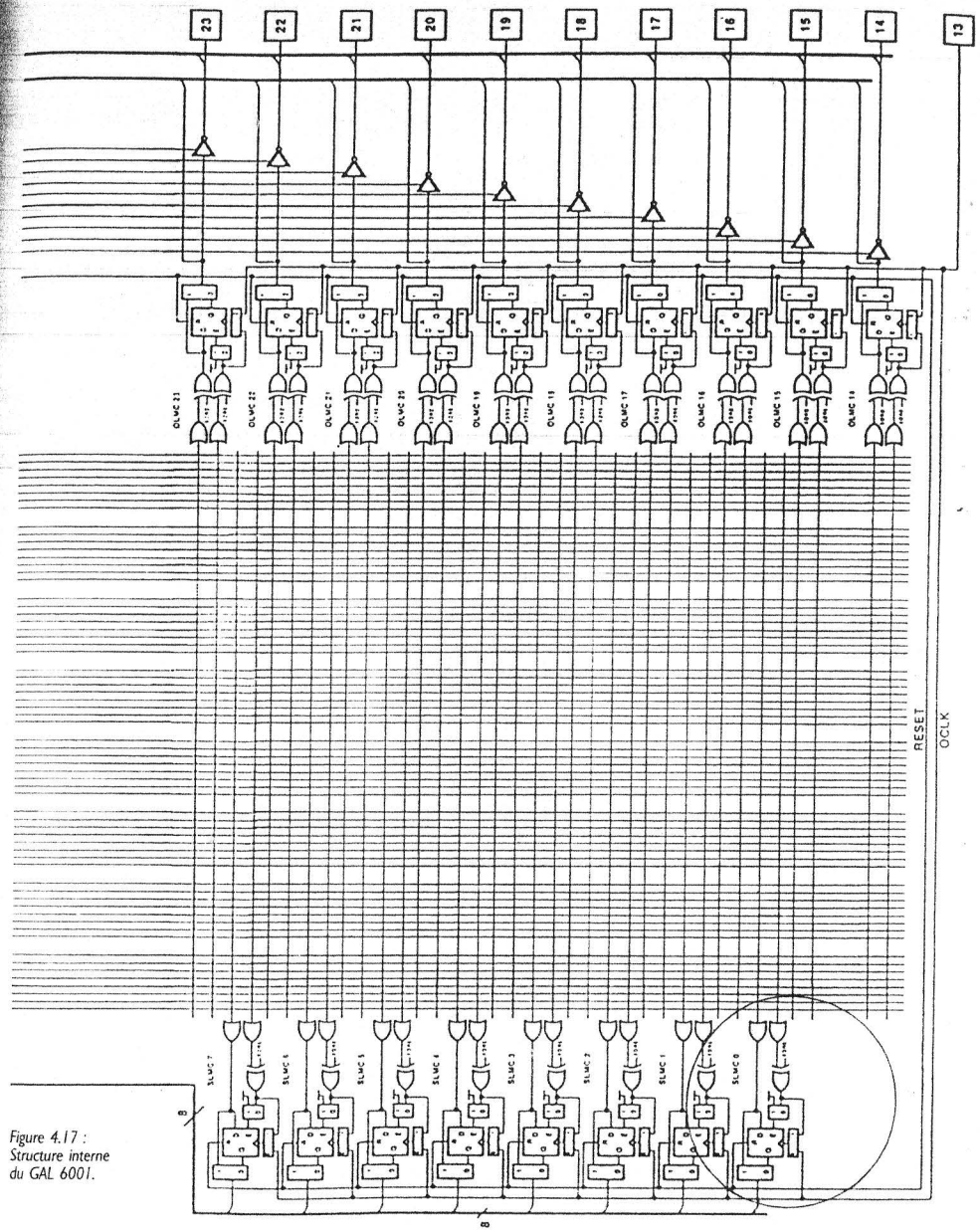


Figure 4.17 : Structure interne du GAL 6001.

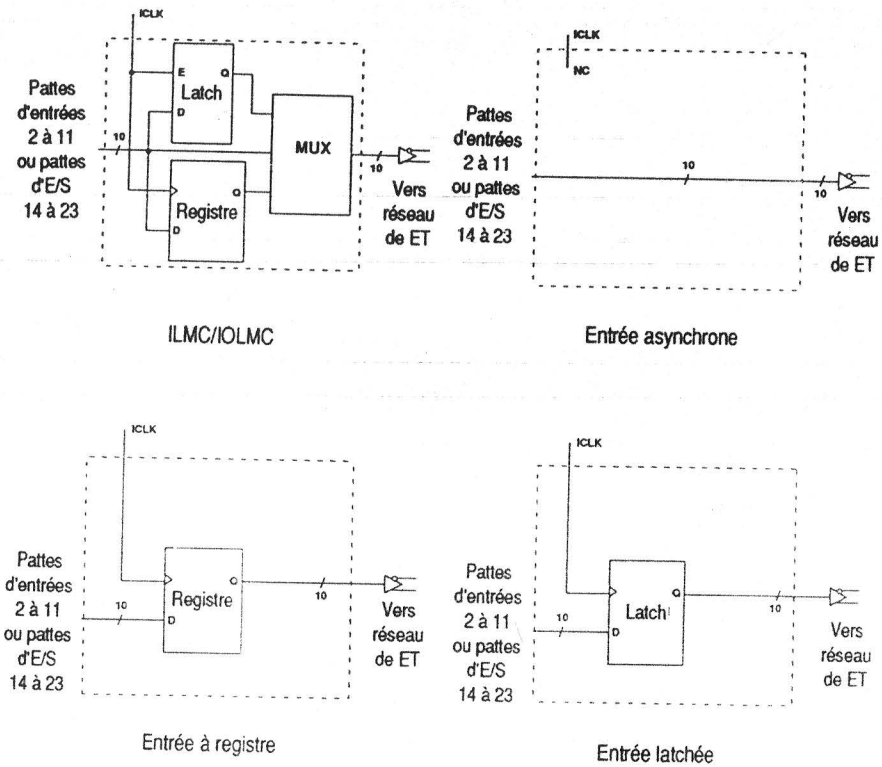


Figure 4.18 :
La macro-cellule d'entrée d'un GAL 6001 et ses diverses configurations.

La zone de ET d'entrées est organisée en 78 lignes d'entrées sur 75 sorties de termes produits. Les 10 ILMC, les 10 IOLMC, les 8 retours des BLMC, les 10 retours des OLMC et la ligne d'entrée d'horloge donnent en effet 39 signaux qui sont tous disponibles en direct et en inverse ce qui donne bien un total de 78. La zone de OU de sorties n'est pas en reste, car elle dispose de 64 entrées et de 36 sorties sommes, ce qui confère au circuit une souplesse de programmation exceptionnelle.

Ce circuit dispose bien évidemment de toutes les facilités offertes par les autres GAL : effacement rapide, bit ou cellule de sécurité, etc.

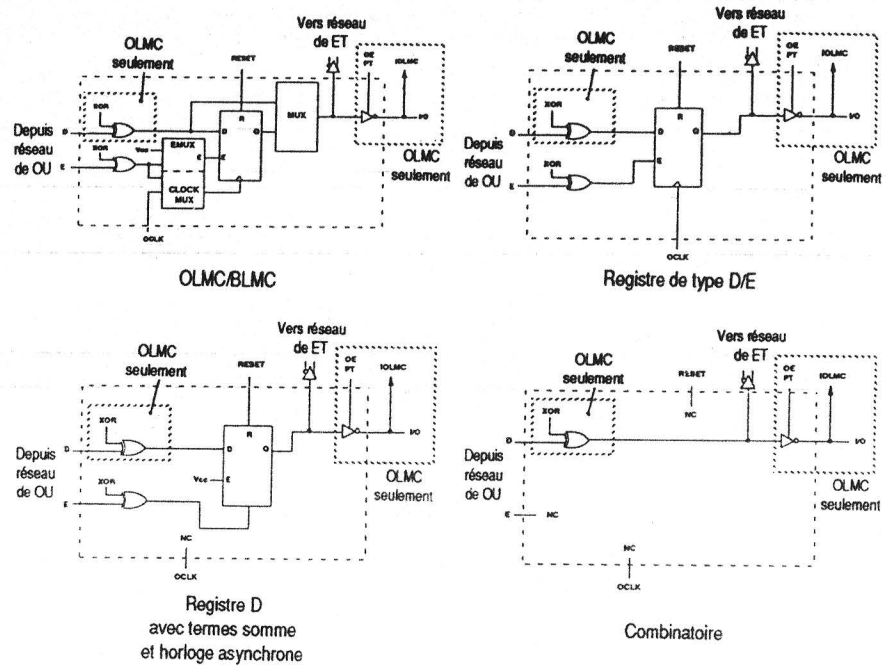


Figure 4.19 :
Les macro-cellules de sortie et enterrées d'un GAL 6001 et leurs diverses configurations.

Comme les PAL CMOS et les GAL, il dispose aussi d'une circuiterie de reset automatique de tous les registres à la mise sous tension lui permettant de démarrer dans un état toujours parfaitement défini. Rappelons que cette fonction place toutes les sorties Q de tous les registres à zéro. Enfin, le GAL 6002 que nous avons superbement ignoré depuis le début de cet exposé est une version plus rapide du 6001, mais adoptant la même architecture interne. Il ne nécessite donc pas de développement spécifique dans le cadre de cet exposé.

Précisons pour conclure ce chapitre que ces GAL n'ont, à ce jour, aucun équivalent chez d'autres fabricants, même chez ceux qui sont déjà producteurs de PAL CMOS.

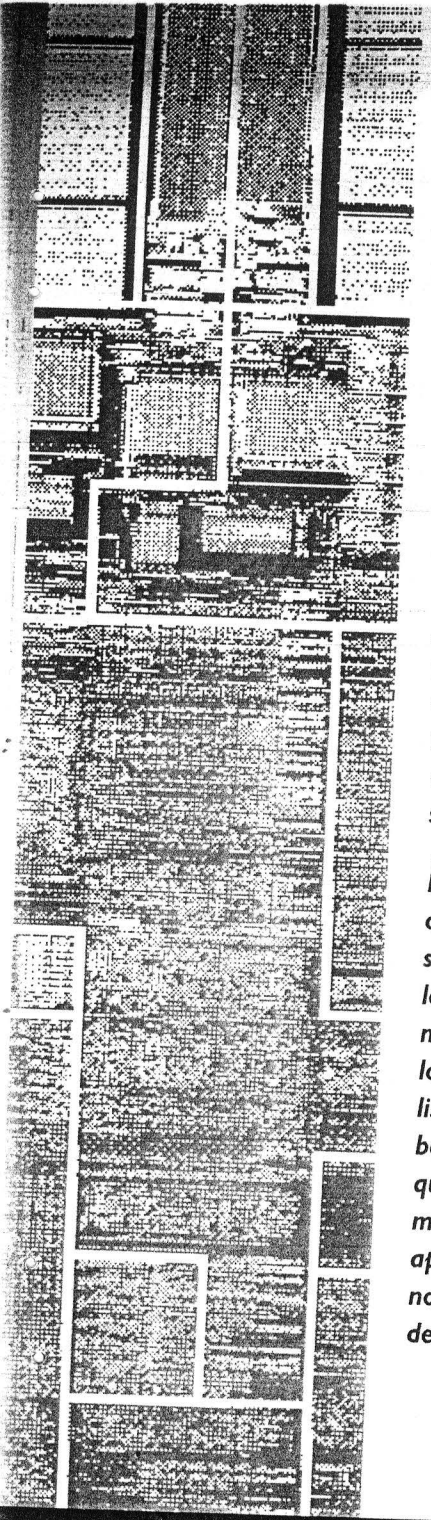
CIRCUITS LOGIQUES PROGRAMMABLES

CHAPITRE 5

Les EPLD.

EPLD, EPLD MAX, ispLSI, MACH, pLSI

- Les EPLD Altera 145
- Les EPLD MACH d'AMD 181
- Les EPLD pLSI et ispLSi de Lattice 190



Comme nous l'avons vu lors de la classification générale présentée chapitre 1, les EPLD sont des circuits logiques programmables électriquement et effaçables, soit par exposition aux ultraviolets pour les plus anciens d'entre eux, soit électriquement. Le terme EPLD signifie, quant à lui, Erasable Programmable Logic Device, c'est-à-dire circuit logique programmable et effaçable.

Arrivé à ce stade de lecture de notre ouvrage, vous pourriez être tenté de croire que les EPLD ne sont jamais qu'une variante des PAL CMOS, puisque les principes de programmation et d'effacement semblent identiques. En fait, il n'en est rien car la structure interne de ces circuits ainsi que les types de macro-cellules et de réseaux d'interconnexion qui y sont employés sont différents.

Les procédés physiques d'intégration étant également différents, les taux d'intégration permis par les EPLD sont nettement plus importants que ceux autorisés par les PAL CMOS. En effet, les plus gros EPLD actuellement commercialisés intègrent jusqu'à 24000 portes logiques dont 12000 sont réellement accessibles à l'utilisateur. On peut ainsi loger dans un seul et unique boîtier l'équivalent d'un schéma logique utilisant jusqu'à 50 à 100 PAL classiques ou bien encore jusqu'à un millier de fonctions TTL ordinaires. Ces chiffres, pour approximatifs qu'ils soient, montrent néanmoins que nous franchissons avec ces produits une étape dans la densité d'intégration offerte à l'utilisateur.

Les EPLD.

EPLD, EPLD MAX, ispLSI, MACH, pLSI

Au risque de nous répéter, et parce que les chiffres que nous venons de citer peuvent sembler impressionnants, nous insistons bien à nouveau sur le fait que, comme pour tous les circuits présentés dans ce livre, il s'agit de boîtiers entièrement programmables par vos soins avec un matériel et des logiciels simples et à la portée de n'importe quel petit laboratoire ou petite entreprise, tant leur coût est réduit.

Les EPLD tels que nous venons de les présenter sommairement ont été inventés par la firme américaine Altera qui les a introduits pour la première fois sur le marché en 1984. Ces circuits sont donc arrivés à une époque où n'existaient guère que les PAL classiques à fusibles pour réaliser de la logique programmable. En raison de leur densité d'intégration et de leur possibilité d'effacement, ils ont connu immédiatement un très vif succès surtout auprès des laboratoires d'études.

Depuis, le concept a été repris par d'autres et on se doit de citer aussi les circuits MACH d'AMD qui utilisent le même concept que les EPLD les plus récents d'Altera. La gamme proposée par AMD étant moins riche que celle d'Altera et étant arrivée après, elle sera présentée en fin de ce chapitre, les notions de base vues pour les EPLD Altera étant en partie utilisables pour les circuits MACH comme vous le constaterez au moment opportun.

La firme Lattice, dont nous avons largement parlé lors de la présentation des GAL, dispose également d'une gamme de circuits appelés pLSI pour programmable LSI et ispLSI pour les versions de ces mêmes composants programmables en circuit. Bien que ces pLSI et ispLSI aient une architecture intermédiaire entre les EPLD et les FPGA, nous avons estimé plus logique de vous les présenter en fin de ce chapitre, car ils font tout de même appel à de nombreux concepts que vous aurez vus auparavant avec la présentation des EPLD d'Altera.

Parallèlement à l'introduction sur le marché de ses premiers EPLD, la société Altera a eu la très bonne idée de commercialiser un outil de développement et de programmation implantable sur n'importe quel micro-ordinateur compatible PC. Cet outil, peu coûteux à l'époque eu égard aux possibilités offertes, est, depuis l'origine, d'une simplicité d'emploi et d'une convivialité exemplaires qui permettent à n'importe quel concepteur de circuits logiques, quelle que soit sa façon de travailler, de produire un EPLD programmé en quelques heures seulement et quasiment sans phase d'apprentissage. Nous en parlerons un peu plus en détail dans le paragraphe consacré à ce sujet plus avant dans ce chapitre.

LES EPLD ALTERA

GÉNÉRALITÉS

L'introduction des EPLD telle que l'a voulue Altera visait deux buts distincts :

- Permettre une densité d'intégration nettement supérieure à celle offerte avec les PAL et aussi proche que possible de celle permise par les réseaux de portes programmables.
- Fonctionner à une vitesse, sinon égale, du moins comparable à celle des PAL bipolaires et en tout cas nettement supérieure à celle permise par les réseaux de portes traditionnels.

Ces deux buts étaient évidemment associés à l'atout majeur des EPLD lors de leur mise sur le marché : la possibilité d'effacement, effacement qui ne pouvait avoir lieu à l'époque qu'aux ultraviolets. Depuis le premier circuit mis sur le marché en 1984, Altera n'a cessé de travailler dans le sens du respect de ces deux buts et propose maintenant une gamme de produits très diversifiée, reposant sur les mêmes concepts et offrant en outre la possibilité d'effacement électrique. On trouve en effet :

- Les EPLD à usage général ou EPLD classiques de la série EP qui ont pour principale fonction le remplacement de logique conçue avec un ou plusieurs PAL, ou bien encore avec plusieurs dizaines de boîtiers TTL classiques. Ces circuits existent en différentes versions avec des boîtiers disposant de 20 à 68 pattes. Leurs délais de propagation typiques sont de l'ordre de 12 ns et des versions pouvant fonctionner jusqu'à 100 MHz sont disponibles. Ces circuits ne sont effaçables qu'aux ultraviolets.
- Les EPLD de la série MAX 5000 qui ont les mêmes fonctions que leurs homologues de la série EP avec toutefois des densités d'intégration plus élevées et une architecture d'interconnexion différente. Ils permettent en effet d'intégrer des réalisations utilisant de 20 à 25 PAL classiques ou bien encore une centaine de boîtiers TTL ordinaires. Ces EPLD sont affublés de la lettre distinctive M dans leurs références car ils utilisent l'architecture dite MAX (Multiple Array matriX) qui autorise une densité d'interconnexions internes plus élevée que l'architecture employée pour les EPLD classiques. Ces circuits ne sont également effaçables qu'aux ultraviolets.

- Les EPLD de la série MAX 7000, plus récents et plus denses que les MAX 5000. Ils généralisent la méthode d'interconnexion originale employée dans les MAX 5000 et apportent en plus la possibilité d'effacement électrique.
- Les EPLD de la série MAX 9000 enfin, encore plus denses et plus riches en termes de possibilités d'interconnexions que les précédents avec toujours, l'effacement électrique mais aussi la programmation en circuit sous une tension unique de 5 volts.

Altera commercialise aussi une gamme de produits appelée famille FLEX. Bien qu'étant des EPLD, ces circuits font appel pour la programmation des interconnexions, à un plan mémoire RAM *enterré* sous la logique, à la manière des LCA présentés au chapitre suivant. Nous avons donc estimé logique de vous présenter également cette famille FLEX au chapitre 6.

Tous ces circuits, quelle que soit leur famille d'appartenance, sont disponibles dans divers types de boîtiers dépendant du nombre d'entrées/sorties disponibles. Actuellement, ces boîtiers vont de 20 pattes à plus de 300 pattes pour les plus gros EPLD MAX 9000. On trouve des modèles DIL classiques jusqu'à 40 pattes et des modèles LCC, PGA ou flat pack pour les nombres de pattes supérieurs.

Les versions effaçables aux ultraviolets des EPLD sont évidemment proposées dans des boîtiers à fenêtre afin de pouvoir exposer la puce aux ultraviolets. Comme on ne sait pas placer de fenêtre dans des boîtiers plastiques, ceux-ci sont nécessairement en céramique.

Par contre, compte tenu du nombre de produits développés avec succès avec des EPLD qui ne nécessitent plus ensuite d'être effacés, tous les circuits existent aussi en version sans fenêtre qui peut alors être intégrée dans un boîtier plastique afin de réduire les coûts. On se trouve alors en présence d'EPLD de type OTP (*One Time Programmable*) exactement comme les mémoires UVPROM sans fenêtre vues au chapitre 2 et appelées elles aussi OTP (pour *One Time Prom* il est vrai, mais la variante d'appellation est légère et justifiée).

Les EPLD EECMOS, quant à eux, sont évidemment toujours proposés en boîtiers sans fenêtre et ne doivent donc pas être confondus avec des modèles OTP.

LES CELLULES LOGIQUES DE BASE DES EPLD ALTERA

Comme les PAL CMOS et les GAL, les EPLD font appel à la notion de macro-cellule qui permet, par programmation, de réaliser de nombreuses fonctions logiques de base.

Le schéma-type de la macro-cellule de tous les EPLD vous est présenté figure 5.1 dans sa version détaillée correspondant à sa réalisation réelle au sein du circuit. La figure 5.2, quant à elle, présente les différents sous-ensembles susceptibles d'être réalisés, ensemble ou séparément les uns des autres, par cette macro-cellule. Nous allons commenter quelque peu ces deux schémas afin de vous faire prendre conscience de la souplesse de programmation ainsi offerte à l'utilisateur.

Remarquez dès à présent que sur les EPLD les plus denses et les plus récents, cette cellule évolue peu quant à ses possibilités logiques ; par contre, le nombre d'entrées et de sorties et leurs possibilités d'interconnexions changent selon que l'on fait appel à la famille MAX 5000, 7000 ou 9000. Nous verrons cela au moment opportun.

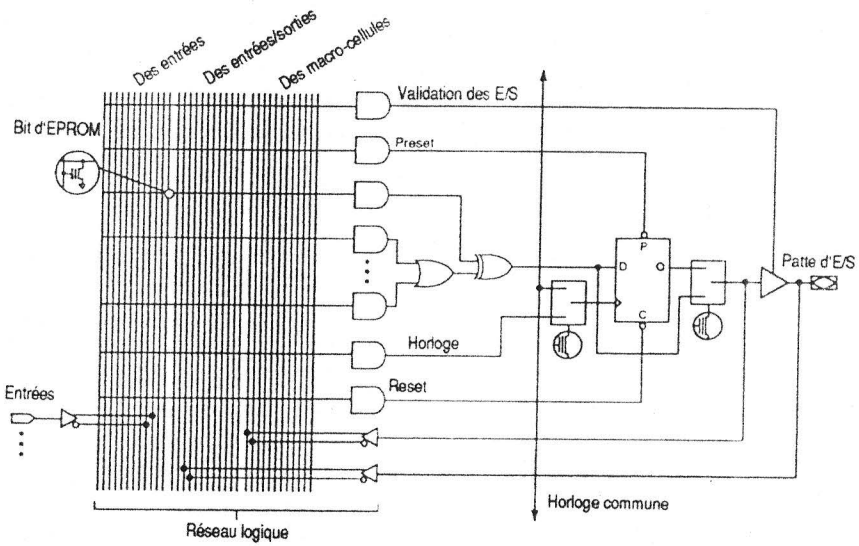


Figure 5.1 :
Schéma type de la macro-cellule d'un EPLD.

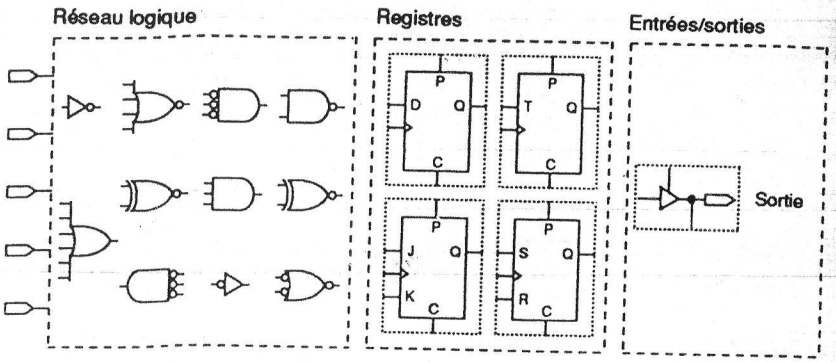


Figure 5.2 :
Les sous-ensembles réalisables avec la macro-cellule d'un EPLD.

Intéressons-nous pour l'instant à la zone de connexions programmables présentée sur la partie gauche de la figure 5.1. Nous y retrouvons le même principe de matrice de liaison que celui vu pour les PAL. Par contre, les signaux disponibles sont groupés en trois blocs distincts selon leur provenance qui peut être : des signaux d'entrées provenant de pattes réservées à cet usage et appelées entrées dédiées, des signaux d'entrées/sorties et des signaux provenant d'autres macro-cellules. Les possibilités d'interconnexion sont ainsi très nombreuses.

Au centre de cette macro-cellule se trouve une bascule qui peut, par programme, prendre l'une des configurations présentées figure 5.2. En d'autres termes, cette bascule peut donc être de type D, de type T, de type R-S ou bien encore de type J-K. Autant dire que toutes les formes usuellement employées en logique conventionnelle sont disponibles. Bien sûr, si cette bascule est inutile compte tenu de la fonction à réaliser, elle peut être éliminée grâce à la programmation du multiplexeur visible sur sa droite en figure 5.1.

L'horloge de cette bascule peut être commune à plusieurs autres et provenir alors d'une ligne spécialisée reliée à une entrée dédiée ne passant pas dans la zone d'interconnexions et permettant ainsi de garantir des temps de propagation et des synchronismes de fonctionnement parfaitement connus. Elle peut aussi provenir de la zone programmable et résulter alors de n'importe quelle combinaison de signaux. Le choix se fait ici encore par programmation grâce au multiplexeur visible à gauche de la bascule sur la figure 5.1.

Si l'horloge est commune, la bascule change d'état sur un front montant. Si, par contre, l'horloge provient de la zone programmable, la bascule peut être programmée pour changer d'état sur un front montant ou descendant.

Cette bascule peut être pré-positionnée à 0 ou à 1 au moyen de signaux provenant eux aussi de la zone programmable. Enfin, la donnée appliquée à son entrée D (ou la donnée disponible en sortie de la macro-cellule si la bascule a été invalidée) est une combinaison de signaux provenant de la zone programmable. Remarquez la présence d'un OU Exclusif permettant la réalisation très facile de logique de polarités opposées puisqu'un OU Exclusif n'est autre qu'un inverseur-non inverseur commandé par un signal logique.

La sortie de cette bascule est ré-injectée sur la zone programmable afin de permettre des rebouclages ou, tout simplement, afin de permettre son envoi vers d'autres macro-cellules. Elle est en outre appliquée à un buffer trois état validé par un signal en provenance de la zone programmable.

La patte du boîtier reliée à ce buffer pouvant être une entrée/sortie, elle se trouve elle aussi envoyée sur la zone programmable pour exploitation en tant qu'entrée.

Afin que cela soit bien clair, la figure 5.3 présente juste la structure d'entrée/sortie d'un EPLD sur laquelle on distingue bien les différents signaux et leurs fonctions :

- La validation du buffer de sortie en provenance de la ou des macro-cellules.

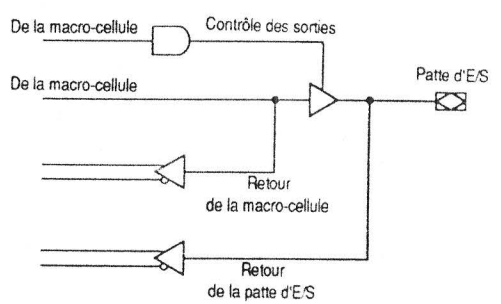


Figure 5.3 :
Détail de la partie entrées/sorties de la macro-cellule d'un EPLD.

- La sortie de la macro-cellule proprement dite.
- Le retour ou feedback de la macro-cellule proprement dite.
- Le retour ou feedback de la sortie proprement dite qui est aussi le signal d'entrée si la patte correspondante est utilisée en tant que telle.

Attention ! Cette structure se retrouve sur toutes les pattes d'entrées/sorties des EPLD. Par contre, rien de tout cela n'est disponible sur les pattes d'entrées dédiées qui sont donc directement dirigées sur la zone programmable, sans plus.

LES PARTICULARITÉS TECHNOLOGIQUES

Les EPLD sont des circuits réalisés en technologie CMOS, ce qui implique certains avantages, mais aussi, hélas, certaines limitations.

L'intérêt majeur de la technologie CMOS est sa consommation particulièrement faible qui permet, surtout dans le cas des EPLD, des densités d'intégration particulièrement élevées. Il est en effet plus facile de placer sur une puce 1000 portes qui consomment chacune 200 µW que le même nombre de portes réalisées en technologie bipolaire par exemple, et qui consommeraient 1 mW par porte.

Comme vous le savez sans doute, les circuits CMOS présentent la particularité de voir leur consommation augmenter avec la fréquence de travail. Ceci résulte principalement de ce que la seule énergie à fournir est celle nécessaire pour vaincre les charges et décharges des capacités parasites et que ceci augmente donc naturellement avec la fréquence.

De ce fait, Altera propose deux modes de travail distincts pour ses EPLD. Un mode ultra faible consommation baptisé *zero-standby*, ce qui signifie que le circuit ne consomme rien lorsqu'il est au repos. Dans ce mode, librement choisi par vos soins au moment de la programmation du circuit, chaque entrée se trouve reliée à un détecteur de transition. Tant que ce détecteur ne reçoit pas d'information valide, donc tant que le circuit est laissé au repos par son entourage, la consommation interne se chiffre en µA voir même moins selon les boîtiers. Par contre, dès que ce détecteur décèle une transition, il *réveille* le circuit qui se met alors à fonctionner normalement.

Ce processus est très efficace mais, bien évidemment, introduit une augmentation du temps de transfert entrée/sortie de l'ordre de 30 à 40 %. Ce n'est pas forcément pénalisant pour certaines applications où la consommation est un critère primordial.

Par contre, si la vitesse est le paramètre essentiel de votre application, il est possible de désactiver cette fonction par programmation. On passe alors l'EPLD dans le mode *turbo*, (mais non, nous n'avons rien inventé, Altera l'appelle comme ça) qui est systématiquement proposé par le logiciel de programmation du circuit. Dans ce mode, les circuits fonctionnent alors à la vitesse maximum permise par leur technologie.

Un autre point essentiel à évoquer lié à la technologie utilisée est celui de la durée de rétention des informations. En effet, si les EPLD n'étaient utilisés qu'en phase de développement de produits c'est-à-dire au stade du laboratoire, des prototypes ou des maquettes, ce paramètre n'aurait pas grande importance. Mais, comme du fait de l'intérêt des EPLD et de leur grande densité d'intégration, des versions OTP sont placées dans des produits finis, ce paramètre devient essentiel.

Altera indique donc effectuer des *burn-in* de ses circuits pendant plusieurs semaines à 250 °C, ce qui équivaut à un fonctionnement continu du circuit à 70 °C pendant 1000 ans sans perte d'information. On peut donc être tranquille de ce côté.

Le dernier point à signaler concerne les protections qu'il peut être nécessaire de prévoir sur les pattes des EPLD dans quelques cas particuliers. En effet, dans certaines applications, il est parfois nécessaire de pouvoir mettre en place des cartes, sur un bus par exemple, alors même que la machine est sous tension. Si les circuits bipolaires acceptent relativement bien ce traitement, il est en général destructeur pour les circuits CMOS qui ne tolèrent pas qu'on leur applique des signaux alors qu'ils ne sont pas alimentés.

Si donc une ou plusieurs pattes d'un EPLD se trouvent ramenées directement sur un point de connexion susceptible de subir ce traitement (c'est-à-dire en général sur un point de connecteur de bus), il est conseillé de prévoir la protection schématisée figure 5.4. On le voit, ce n'est ni très encombrant, ni très coûteux, surtout que

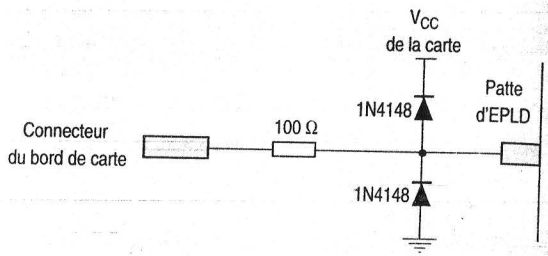


Figure 5.4 :
Protection des pattes exposées d'un EPLD.

cela n'est à faire, répétons-le, que sur les pattes exposées au monde extérieur.

Dernière précision mais qui a son importance : comme tous les circuits programmables vus jusqu'à présent, les EPLD disposent d'une cellule de sécurité qui, une fois programmée, interdit toute relecture du circuit et donc toute copie non autorisée. Comme l'effacement d'un EPLD, qu'il soit électrique ou à ultraviolets, est forcément global, la sécurité apportée par cette cellule est totale.

LES OUTILS DE DÉVELOPPEMENT

Les produits Altera sont évidemment supportés par de nombreux logiciels standards tels ceux déjà cités pour les PAL et les GAL. Néanmoins, et comme nous l'avons laissé entendre en introduction, une partie de ce qui a fait la force des produits Altera a été la commercialisation conjointe d'un logiciel de développement particulièrement souple et puissant. Ce logiciel a maintenant considérablement évolué et existe en de nombreuses versions fonctionnant sur divers types de micro-ordinateurs dont bien sûr les compatibles PC et les stations de travail classiques.

Sans vouloir entrer dans le détail des fonctions de chacun d'entre eux car ce n'est pas notre but, voici tout de même quelques notions essentielles sur ces produits et sur leur mode de travail. Nous avons choisi à titre d'exemple la version pour compatible PC, mais ce que nous écrivons reste bien évidemment vrai pour les versions destinées aux stations de travail.

Chaque ensemble de développement est composé d'un ou plusieurs logiciels et d'une carte à enficher dans un connecteur d'extension ou slot libre du PC. Cette carte ne sert qu'à la programmation physique du circuit. Elle est reliée par un câble plat à un boîtier externe qui reçoit divers adaptateurs pour les supports des différentes versions d'EPLD.

Le logiciel travaille en mode graphique et utilise la technique désormais bien connue des menus déroulants pilotés par les flèches de déplacement du curseur ou, ce qui est incomparablement plus souple, par la souris. Il se compose de plusieurs modules que l'on peut décrire simplement de la façon suivante pour les trois principaux d'entre eux :

- Un module d'entrée qui permet de saisir directement la description de la fonction logique à réaliser sous diverses formes mais aussi et surtout sous forme de schéma classique.
- Un module compilateur qui traduit ce schéma, l'analyse, cherche à le réduire en appliquant les règles logiques classiques et essaye ensuite de le faire rentrer dans le plus petit EPLD possible.
- Un module programmateur qui, à partir des informations générées par le compilateur, peut programmer directement le circuit choisi.

Ces trois modules s'enchaînent naturellement au point que, si votre schéma logique est au point et si le circuit dans lequel le compilateur a réussi à le placer vous convient, il ne faut que de quelques minutes à une ou deux heures pour disposer du boîtier programmé.

Divers outils annexes facilitent l'exploitation de ce système de développement dont les deux plus importants sont le simulateur et la bibliothèque de macro-fonctions.

Cette dernière est destinée à faciliter le travail de conception de votre circuit logique. En effet, elle vous propose un certain nombre de sous-ensembles fonctionnels classiques (compteurs, additionneurs, diviseurs programmables, multiplexeurs) qui ont déjà été réalisés et compilés de la façon la plus efficace qui soit, en termes d'EPLD, par les ingénieurs d'Altera. Vous pouvez alors directement intégrer ces macro-fonctions dans votre schéma. Si cela ne vous suffit pas et que vous êtes un inconditionnel de la logique traditionnelle, sachez aussi que cette bibliothèque vous propose une centaine de références TTL

classiques (74138, 7490, etc.) qui ont, elles aussi, été compilées au préalable par les ingénieurs d'Altera. Vous pouvez ainsi construire votre schéma avec des circuits TTL ; le logiciel, par appel à cette bibliothèque, le traduira alors automatiquement en fonction des macro-cellules disponibles dans les EPLD.

Nous nous sommes un petit peu étendu sur cette approche car elle est d'une extrême souplesse d'emploi, tant pour les concepteurs qui n'ont pas l'habitude de la logique programmable et qui peuvent ainsi dessiner leurs circuits comme par le passé avec une certitude de réussir leur EPLD, que pour la mise en EPLD d'un schéma logique existant qui a déjà été développé et testé en logique conventionnelle. Dans ce dernier cas, la réduction de taille permise par le passage en EPLD est en général impressionnante et convainc les plus sceptiques de l'intérêt de ces produits.

Le deuxième *accessoire* essentiel de ces outils de développement est le simulateur. Celui-ci permet de vérifier le comportement du circuit réalisé tant au plan fonctionnel qu'au plan temporel. Comme tous les logiciels de ce type, il permet la mise en place de sondes en des points variés du circuit, et même à l'intérieur, bien sûr, ce qui facilite les opérations de détection des problèmes de fonctionnement.

Pour ce qui est du compilateur, nous avons dit qu'il essayait de *faire rentrer* le schéma dans le plus petit EPLD possible. Ceci est évidemment une option prise dans l'intérêt pécuniaire de l'utilisateur, mais il est tout à fait possible de spécifier le boîtier de son choix. Le compilateur utilisera alors cette indication, et le seul risque qui existe est que vous ayez choisi un boîtier trop petit, auquel cas il vous le signalera.

Précisons aussi que vous pouvez également spécifier le brochage qu'aura votre circuit, totalement ou partiellement. Si vous ne le faites pas, le compilateur choisira seul, là encore, mais en optimisant celui-ci compte tenu des interconnexions internes.

Nous en resterons là du court descriptif de ces programmes dont il faut retenir à notre avis qu'ils placent l'EPLD à la portée de tous, et ce, sans nécessiter d'apprentissage particulier. En effet, un utilisateur ne connaissant pas du tout le produit apprend à se servir de l'essentiel des fonctions du logiciel en une journée au maximum et, dès cet instant, il est capable de produire des EPLD fonctionnels.

Si vous souhaitez vous forger une opinion plus précise à leur sujet, nous ne saurions trop vous conseiller d'assister à une démonstration, soit lors d'un salon professionnel, soit lors de celles régulièrement organisées par les distributeurs Altera.

LES EPLD DE LA SÉRIE EP OU EPLD CLASSIQUES

Le paragraphe précédent vous a permis de découvrir les EPLD et vous a fourni toutes les informations utiles à leur sujet. Il nous semble cependant intéressant de vous présenter rapidement quelques boîtiers typiques des principales familles d'EPLD afin que vous puissiez avoir une idée un peu plus précise de ce qui est actuellement disponible sur le marché.

La figure 5.5 permet de faire un rapide survol des différents produits de la famille des EPLD série EP. Sont indiqués tout à la fois le nombre de macro-cellules, le nombre d'entrées/sorties et le temps de propagation typique. Comme vous pouvez le constater, et bien que le nombre de références semble assez limité, le choix est en fait très vaste entre le *petit* EPLD EP 610 avec ses 16 macro-cellules et le *gros* EPLD EP 1810 avec ses 48 macro-cellules. Remarquez la présence d'un faux EPLD avec le EP22V10 qui est évidemment compatible des PAL CMOS et GAL 22V10 vus au chapitre précédent.

	EP22V10 EP22V10E	EP610 EP610T	EP610I	EP910 EP910T	EP910I	EP1810 EP1810T
Nombre de portes	400	600	600	900	900	1 800
Portes utilisables	200	300	300	450	450	900
Macro-cellules	10	16	16	24	24	48
Nombre maximum d'E/S	22	20	20	36	36	64
t_{PD} (ns)	7,5	15 (35)	10	30	12	20 (45)
F max en compteur (MHz)	111,1	83,3 (28,6)	100	33,3	100	50 (22,2)

Figure 5.5 :
Les différents EPLD de la série EP ou EPLD classiques.

La figure 5.6, quant à elle, vous présente les brochages typique d'un EP 610, tandis que la figure 5.7 montre la structure interne du circuit. Les gros blocs rectangulaires sont évidemment les macro-cellules dont nous vous avons présenté le schéma figure 5.1. Dans un EP 610, ils

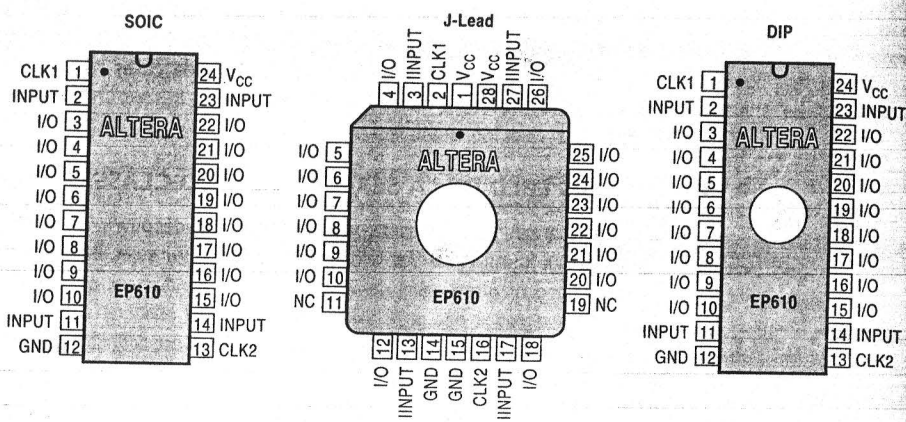


Figure 5.6 :
Brochages typiques des différents boîtiers d'un EP 610.

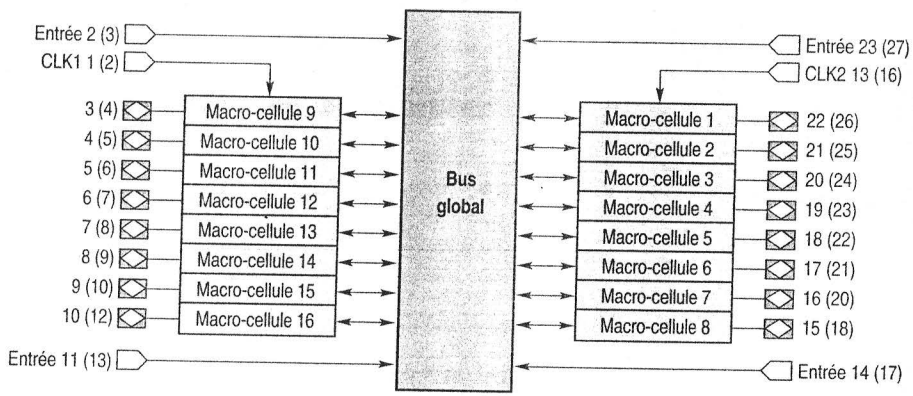


Figure 5.7 :
Structure interne d'un EP 610.

sont au nombre de 16 et correspondent évidemment aux 16 pattes d'entrées/sorties. Remarquez, dans les angles de cette figure, les pattes qui ne sont que des entrées simples ou dédiées et qui, comme nous l'avons dit précédemment, aboutissent donc directement sous forme normale et inversée dans la zone programmable.

Afin que vous ayez une idée plus précise des performances temporelles et fréquentielles d'un tel circuit, le tableau et le chronogramme de la figure 5.8 donnent les temps typiques de la version la plus rapide actuellement disponible.

Le deuxième circuit qu'il nous semble important de vous présenter est l'EP 1810, car il appartient à l'autre extrémité de la gamme, c'est-à-dire celle des plus hautes densités d'intégration de la famille EP s'entend. De ce fait, et bien qu'il respecte tout ce que nous avons expliqué jusqu'à présent, il doit faire appel à une structure interne un peu particulière pour permettre une utilisation efficace de ses macro-cellules. Pour fixer les idées, précisons que notre EP 1810 intègre tout de même l'équivalent de 1800 portes logiques élémentaires.

Il contient, comme nous l'avons vu avec le tableau de la figure 5.5, 48 macro-cellules et dispose en outre de 16 lignes d'entrées dédiées. De ce fait il ne peut tenir que dans un boîtier LCC à 68 pattes ou dans un boîtier PGA de la même taille. Les brochages de ces deux boîtiers vous sont présentés figure 5.9.

Toutes les ressources disponibles sont réparties comme le montre la figure 5.10. Les macro-cellules sont réparties en 4 groupes de 12, disposant de leur propre réseau d'interconnexions programmables local ; réseaux qui sont à leur tour reliés à un réseau ou bus global.

En outre, certaines macro-cellules sont dites locales et d'autres globales. Cette distinction tient uniquement aux possibilités de connexions offertes. En effet, comme le montre la figure 5.11 les macro-cellules locales ont accès aux 16 entrées dédiées, aux retours globaux des 4 groupes de 12 macro-cellules et aux retours du groupe de 12 dans lequel elles se trouvent. Par contre, elles ne peuvent ré-injecter leurs signaux de sortie que sur le bus local. Les macro-cellules globales par contre, comme le montre la figure 5.12, ont accès aux mêmes ressources que les macro-cellules locales mais peuvent ré-injecter leurs sorties sur le bus global.

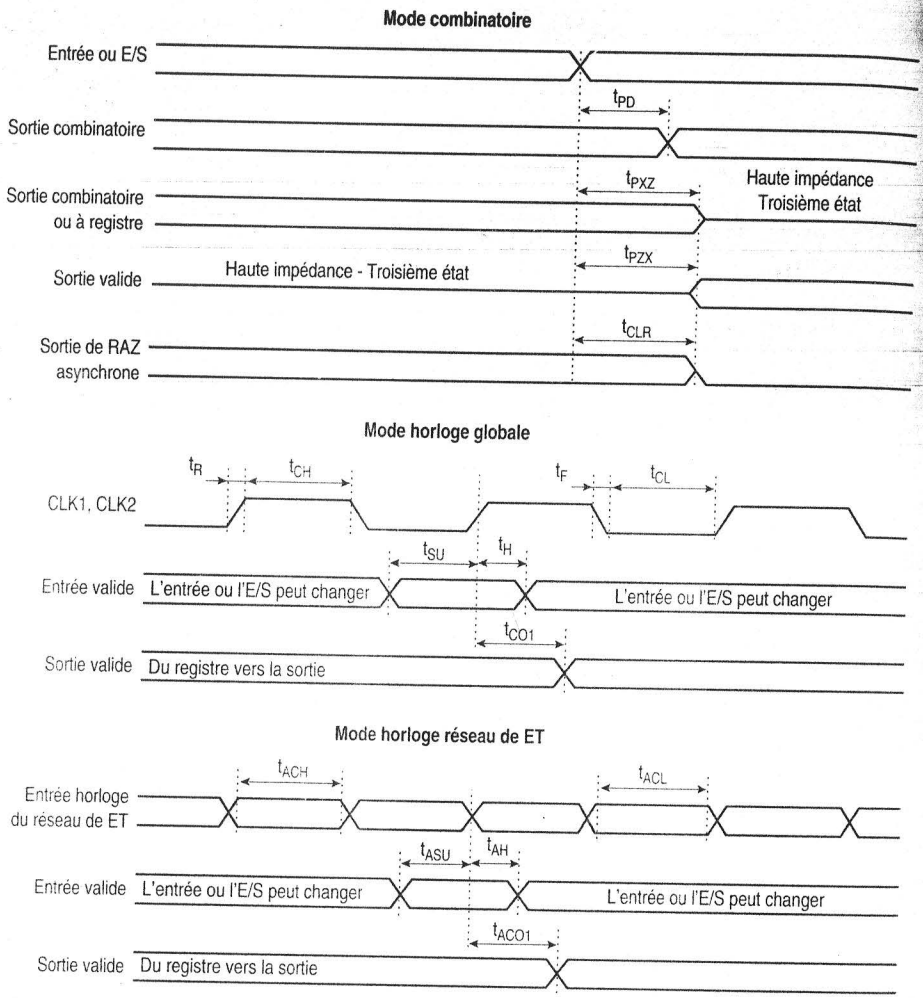
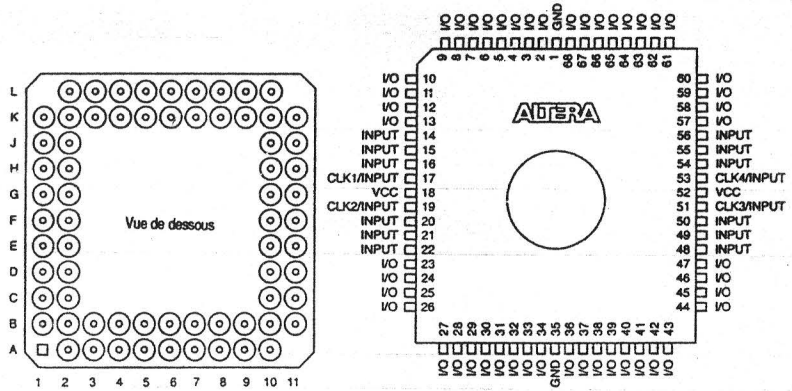


Figure 5.8 :
Caractéristiques de commutation de la version d'EP 610 la plus rapide.

Symbole	Paramètre	Min	Max	Conditions
t_{pd1}	Entrée vers sortie sans registre	–	15 ns	C = 35 pF
t_{pd2}	E/S vers sortie sans registre	–	17 ns	C = 35 pF
t_{pzx}	Entrée vers validation sortie	–	15 ns	C = 35 pF
t_{pxz}	Entrée vers dévalidation sortie	–	15 ns	C = 5 pF
t_{clr}	Temps de RAZ asynchrone	–	15 ns	C = 35 pF
t_{IO}	Délai patte et buffer d'E/S	–	2 ns	
Mode 0 ou mode horloge globale				
f_{max}	Fréquence maximum	83,3 MHz	–	
t_{su}	Temps de setup	9 ns	–	
t_h	Temps de maintien	0 ns	–	
t_{cl}	Temps d'horloge basse	6 ns	–	
t_{ch}	Temps d'horloge haute	6 ns	–	
t_{co1}	Horloge vers sortie	–	11 ns	
t_{cnt}	Période d'horloge minimum	–	12 ns	
f_{cnt}	Fréquence interne maximum	83,3 MHz	–	(1)
Mode 1				
f_{max}	Fréquence maximum	71,4 MHz	–	
t_{asu}	Temps de setup	6 ns	–	
t_{ah}	Temps de maintien	6 ns	–	
t_{acl}	Temps d'horloge basse	7 ns	–	
t_{ach}	Temps d'horloge haute	7 ns	–	
t_{aco1}	Horloge vers sortie	–	15 ns	
t_{acnt}	Période d'horloge minimum	–	14 ns	
f_{acnt}	Fréquence interne maximum	71,4 MHz	–	(1)

(1) Mesuré avec l'EPM 5192 programmé en compteur 16 bits.

Figure 5.8 :
Principales caractéristiques de commutation de la version la plus rapide d'un EPM 5192.



Patte	Fonction	Patte	Fonction	Patte	Fonction	Patte	Fonction
A2	I/O	B9	I/O	F10	GND	K4	INPUT
A3	I/O	B10	I/O	F11	I/O	K5	INPUT
A4	I/O	B11	I/O	G1	I/O	K6	VCC
A5	INPUT	C1	I/O	G2	I/O	K7	INPUT
A6	CLK4/INPUT	C2	I/O	G10	I/O	K8	INPUT
A7	CLK3/INPUT	C10	I/O	G11	I/O	K9	I/O
A8	INPUT	C11	I/O	H1	I/O	K10	I/O
A9	I/O	D1	I/O	H2	I/O	K11	I/O
A10	I/O	D2	I/O	H10	I/O	L2	I/O
B1	I/O	D10	I/O	H11	I/O	L3	I/O
B2	I/O	D11	I/O	J1	I/O	L4	INPUT
B3	I/O	E1	I/O	J2	I/O	L5	CLK1/INPUT
B4	INPUT	E2	I/O	J10	I/O	L6	CLK2/INPUT
B5	INPUT	E10	I/O	J11	I/O	L7	INPUT
B6	VCC	E11	I/O	K1	I/O	L8	I/O
B7	INPUT	F1	I/O	K2	I/O	L9	I/O
B8	INPUT	F2	GND	K3	I/O	L10	I/O

Figure 5.9 :
Les deux boîtiers de l'EP 1810.

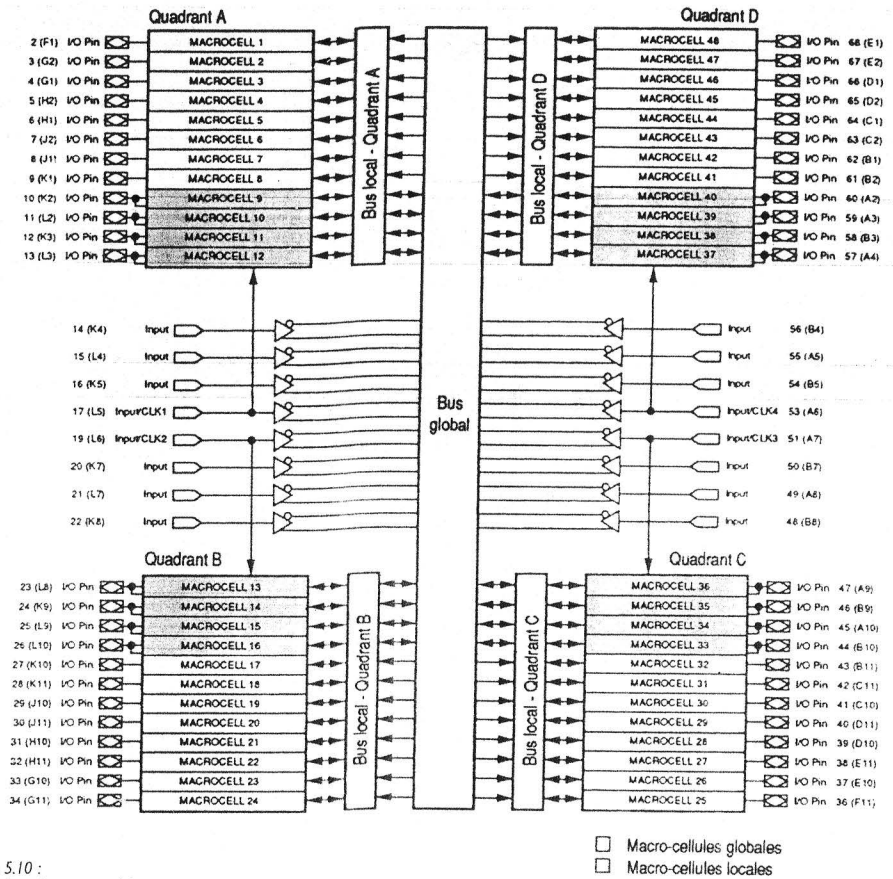


Figure 5.10 :
Répartition des macro-cellules dans un EP 1810.

Bien sûr, nous direz-vous, tout ceci ne concerne pas directement l'utilisateur puisque le placement des fonctions logiques dans le circuit est réalisé par le logiciel de compilation du système de développement. Il est cependant important d'être conscient de cette nécessité qu'il y a de répartir les ressources car, lorsque l'on utilise les circuits de ce type, on s'aperçoit très vite que l'on n'arrive jamais à utiliser à

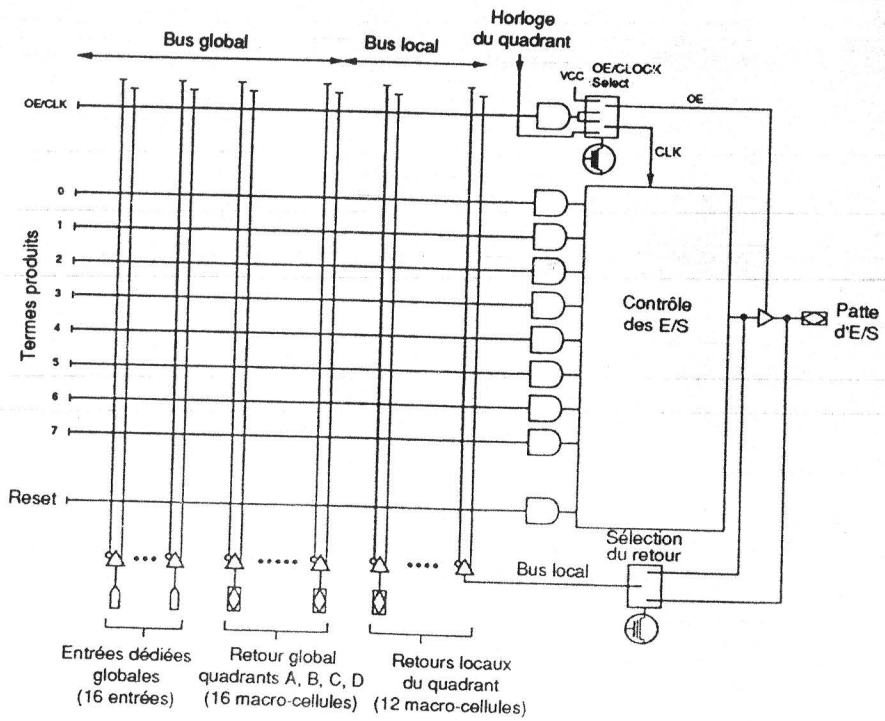


Figure 5.11 :
Les possibilités de connexion des macro-cellules locales.

100 % les ressources internes pour cause de manque de place pour passer certaines interconnexions. Si le taux d'utilisation est voisin de 60 ou 80 %, c'est bien, et même très bien. S'il n'est que de 20 ou 30 %, c'est déjà plus inquiétant et cela peut signifier un mauvais choix de circuit. De ce fait, il est parfois préférable de choisir d'associer deux petits EPLD qui sont utilisés à 80 % plutôt qu'un seul gros qui ne serait employé qu'à 30 % par exemple. Ces essais sont d'autant plus faciles à faire qu'ils ne prennent que quelques minutes de compilation et que le compilateur indique à chaque fois de façon très précise le taux d'utilisation des diverses ressources internes des boîtiers.

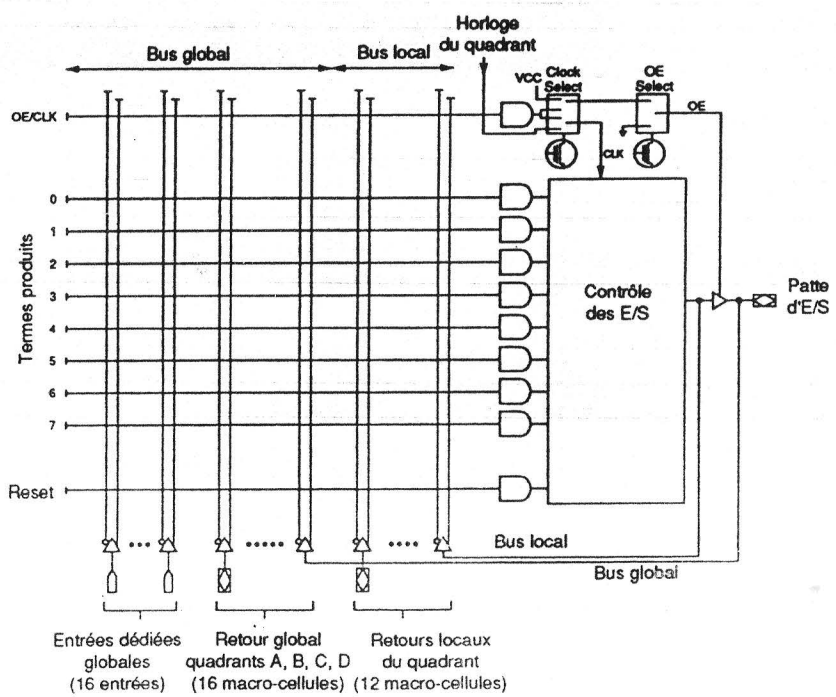


Figure 5.12 :
Les possibilités de connexion des macro-cellules globales.

Ces remarques ont d'ailleurs conduit tous les fabricants de circuits logiques programmables, que ce soit Altera avec ses EPLD, AMD avec ses circuits MACH ou Xilinx avec ses LCA à adopter des méthodes d'interconnexions de plus en plus performantes.

LES EPLD DE LA FAMILLE MAX 5000

Avec ces circuits, nous franchissons une nouvelle étape dans la densité d'intégration offerte par les EPLD ; en effet, alors que le plus gros boîtier de la série EP, l'EP 1810 dont nous venons de parler, contient 1800 portes logiques de base, on arrive ici à près de 7500 portes avec l'EPM 5192.

Le tableau de la figure 5.13 présente les différentes références disponibles dans la série MAX 5000.

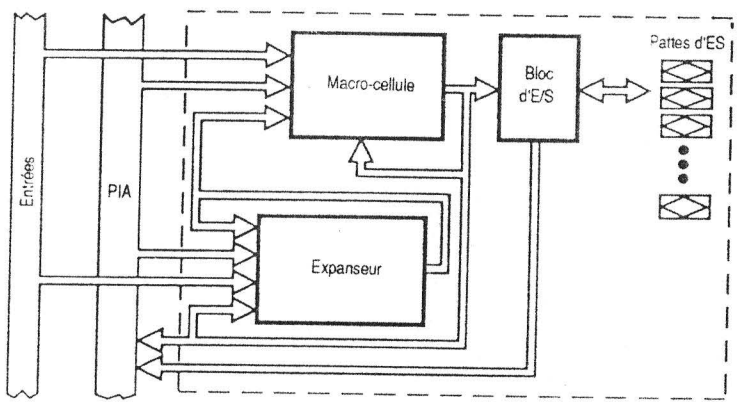
	EPM5032	EPM5064	EPM5128	EPM5130	EPM5192
Nombre de portes	1,200	2,500	5,000	5,00	7,500
Portes utilisables	600	1,250	2,500	2,500	3,750
Macro-cellules	32	64	128	128	192
LAB	1	4	8	8	12
Expanseurs	64	128	256	256	384
Routage	global	PIA	PIA	PIA	PIA
NB maximum d'E/S	24	36	60	68,84	72
t_{PD} (ns)	10	5	15	15	15
t_{ASU} (ns)	3	5	5	5	5
t_{CO} (ns)	6	8	8	8	8
F max en compteur (MHz)	125	83.3	83.3	83.3	83.3

Figure 5.13 :
Les différents EPLD de la famille MAX 5000.

Le concept de la série MAX est analogue à celui des autres EPLD, largement exposé précédemment, mais pour diminuer l'influence des problèmes d'interconnexions internes que nous venons d'évoquer, une approche particulière a été utilisée. La macro-cellule de base de l'EPLD reste utilisée dans les circuits MAX ; par contre, cette macro-cellule est intégrée dans ce qu'Altera appelle un LAB ou Logic Array Block. Chaque circuit de la série 5000 contient ainsi de 1 à 12 LAB.

Figure 5.14 :
La structure LAB utilisée dans les EPLD de la famille MAX 5000.

Comme le montre la figure 5.14, un LAB est en fait le regroupement de la macro-cellule avec son registre programmable dont nous avons



vu le schéma détaillé figure 5.1, d'une zone logique sans registre qui est en fait un bloc de logique combinatoire classique jouant le rôle d'expanseur, et de la circuiterie d'interfaçage avec les pattes d'entrées/sorties vue figure 5.3.

La zone expanseur respecte le schéma présenté figure 5.15. Comme on le voit, elle permet donc d'effectuer un regroupement logique de signaux selon diverses fonctions combinatoires et permet ainsi de diminuer le nombre d'interconnexions globales nécessaires. Les lignes d'entrées dédiées aboutissent sur la macro-cellule et sur l'expanseur via la zone programmable classique tandis que les sorties de l'expanseur et de la macro-cellule aboutissent sur une zone programmable différente appelée PIA pour *Programmable Interconnect Array* ou réseau d'interconnexions programmables. Cette zone, à son tour, peut être reliée à l'entrée des macro-cellules et des expanseurs offrant ainsi une souplesse de configuration sans égale.

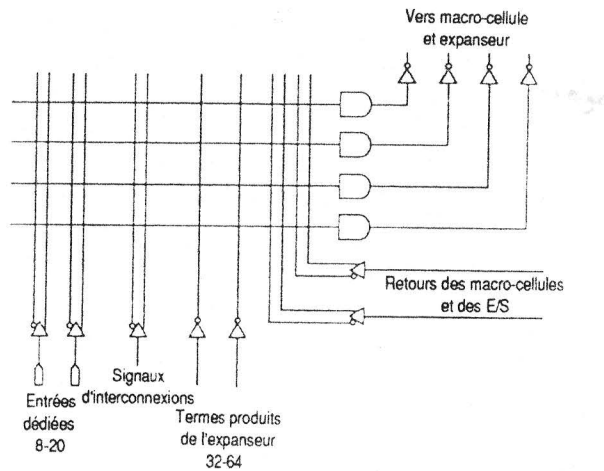


Figure 5.15 :
Principe de l'expanseur utilisé dans les LAB.

Ceci étant précisé, voici à titre d'exemple les informations relatives à l'EPM 5192 qui, avec ses 192 macro-cellules (à comparer aux 48 macro-cellules de l'EP 1810 par exemple) est actuellement le circuit le plus important de la série MAX 5000.

Voici tout d'abord en figure 5.16 le brochage de ce monstre disponible en boîtier LCC à 84 pattes, en boîtier PGA à 84 pattes également et en boîtier flat pack à 100 pattes. Le nombre de pattes est évidemment imposé par le nombre d'entrées/sorties disponibles qui atteint ici 72 entrées ou 64 sorties, ou toute combinaison comprise entre ces deux extrêmes.

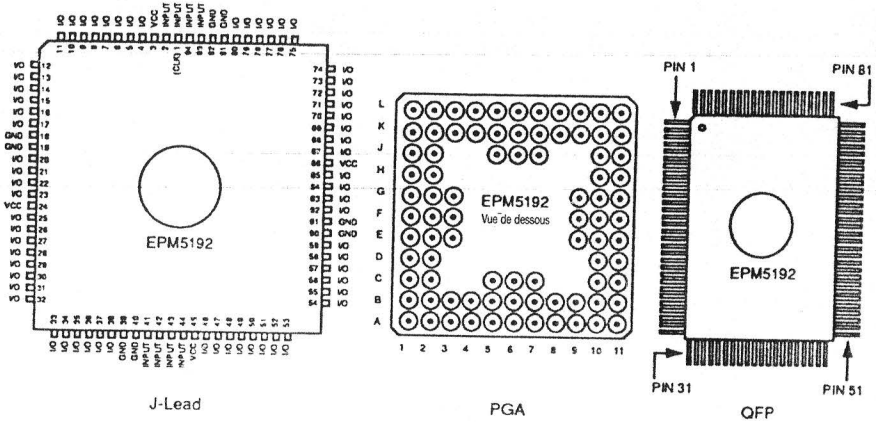


Figure 5.16 :
Les différentes versions de boîtier d'un EPM 5192.

L'organisation interne du circuit vous est présentée figure 5.17. On y reconnaît les diverses macro-cellules groupées au sein de 12 LAB tous reliés à une gigantesque zone d'interconnexion ou PIA. Les entrées dédiées aboutissent, quant à elles, directement sur les LAB comme le montre bien la partie haute de cette figure.

Malgré cette extrême densité d'intégration, l'EPM 5192 reste un circuit rapide comme vous pouvez en juger à l'examen de la figure 5.18 qui présente les données temporelles de la version la plus rapide actuellement disponible. Bien sûr, la fréquence d'horloge garantie n'est *que* de 62,5 MHz pour la version la plus rapide, mais il ne faut pas oublier que cela s'applique à un boîtier dont les possibilités d'intégration sont extrêmement importantes. Il n'est donc pas surprenant que l'on ne soit pas en présence de temps, tels ceux rencontrés dans des PAL n'intégrant, par exemple, que quelques dizaines de portes.

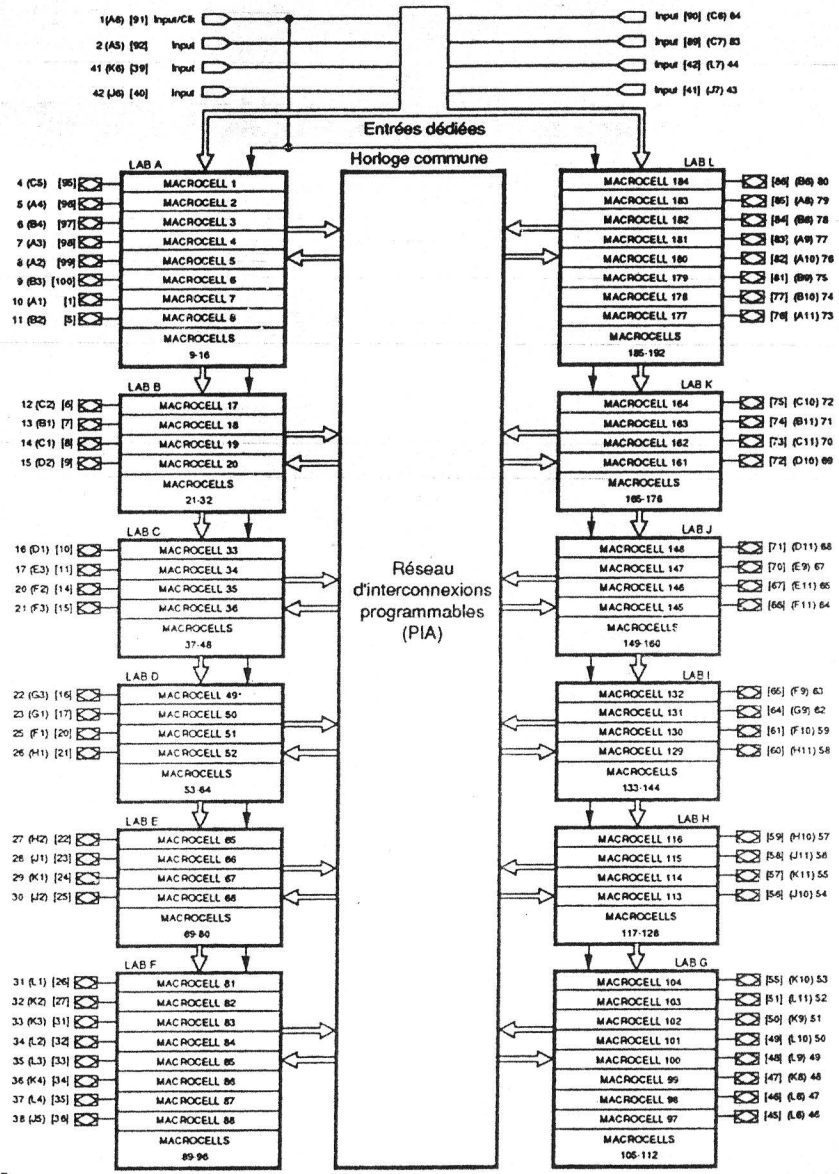


Figure 5.17 :
Organisation interne d'un EPM 5192.

Symbole	Paramètre	Min	Max	Conditions
Paramètres de commutation externes				
t_{pd1}	Entrée vers sortie sans registre	–	25 ns	C = 35 pF
t_{pd2}	E/S vers sortie sans registre	–	40 ns	C = 35 pF
t_{su}	Setup horloge globale	15 ns	–	
t_h	Maintien horloge globale	0 ns	–	
t_{col}	Horloge globale vers sortie	–	14 ns	C = 35 pF
t_{ch}	Temps haut horloge globale	8 ns	–	
t_{cl}	Temps bas horloge globale	8 ns	–	
t_{asu}	Setup horloge réseau	5 ns	–	
t_{ah}	Maintien horloge réseau	6 ns	–	
t_{acol}	Horloge réseau vers sortie	–	25 ns	C = 35 pF
t_{ach}	Temps haut horloge réseau	11 ns	–	
t_{acl}	Temps bas horloge réseau	9 ns	–	
t_{cnt}	Période d'horloge globale minimum	–	20 ns	
f_{cnt}	Fréquence horloge globale maximum	50 MHz	–	(1)
t_{acnt}	Période horloge réseau minimum	–	20 ns	
f_{acnt}	Fréquence horloge réseau maximum	50 MHz	–	(1)
f_{max}	Fréquence d'horloge maximum	62,5 MHz		

Figure 5.18

(1) Mesuré avec l'EPM 5192 programmé en compteur 16 bits.

LES EPLD DE LA FAMILLE MAX 7000

Notre but n'étant pas de transformer cet ouvrage en catalogue Altera, vous vous doutez très certainement que si nous présentons cette famille à part, c'est parce qu'elle apporte quelque chose de résolument nouveau à tout ce que nous avons vu ci-avant.

En fait, les nouveautés sont de trois ordres :

- Les MAX 7000 font appel à une méthode d'interconnexion encore plus souple que celle employée pour les MAX 5000 afin de mieux optimiser l'utilisation des portes internes.
- Ces circuits sont les premiers à être réalisés en technologie EEC-MOS ou E2CMOS, c'est-à-dire qu'ils sont effaçables électriquement comme de vulgaires GAL ou PAL CMOS par exemple.
- Enfin, leur densité d'intégration maximum est également un tout petit peu plus importante comme le montre le tableau de la figure 5.19 qui présente les différentes versions disponibles.

	EPM7032	EPM7032V	EPM7064	EPM7096	EPM7128E	EPM7160E	EPM7192E	EPM7256E
Nombre de portes	1 200	1 200	2 500	3 600	5 000	6 400	7 500	10 000
Portes utilisables	600	600	1 250	1 800	2 500	3 200	3 750	5 000
Macro-cellules	32	32	64	96	128	160	192	256
Nombre maximum d'E/S	36	36	68	76	100	104	124	164
t_{PD} (ns)	5	12	6	6	7,5	7,5	10	10
t_{SU} (ns)	4	10	5	5	6	6	7	7
t_{FSU} (ns)	–	–	–	–	3	3	3	3
t_{CO1} (ns)	3,5	7	4	4	4,5	4,5	5	5
F max en compteur (MHz)	178,6	90,9	151,5	151,5	125	125	100	100

Figure 5.19 :
Les différents EPLD de la famille MAX 7000.

Si les deux dernières nouveautés évoquées ci-avant n'appellent pas d'autres commentaires, il n'en est pas de même de la nouvelle méthode d'interconnexion que nous allons essayer de vous présenter de la façon la plus simple qui soit.

L'architecture générale, visible figures 5.20 et 5.21 selon les références des circuits, utilise toujours la notion de LAB et de PIA vue pour les circuits de la famille MAX 5000. Les macro-cellules sont, par contre, un peu plus complexes comme le montre la figure 5.22.

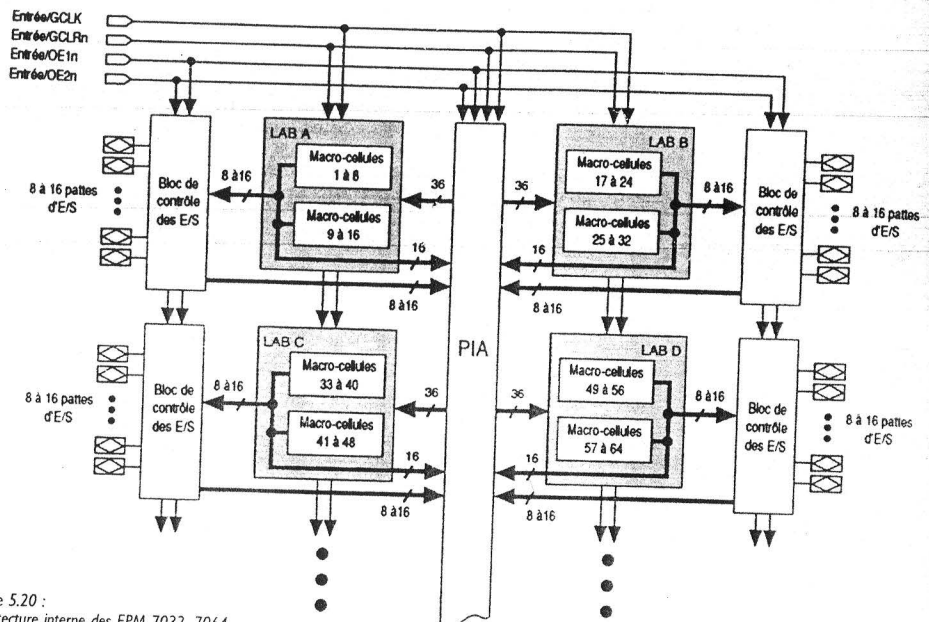


Figure 5.20 :
Architecture interne des EPM 7032, 7064 et 7096.

Les expanseurs, déjà vus pour la famille MAX 5000 offrent également un peu plus de souplesse de programmation ; ainsi, comme le montre la figure 5.23, les termes produits inutilisés dans une macro-cellule peuvent être utilisés par la cellule voisine si nécessaire.

Le contrôle des entrées/sorties enfin a également évolué comme le montrent les figures 5.24 et 5.25 avec une plus grande souplesse de commande du passage en haute impédance du buffer de sortie. Remarquez aussi au niveau de ce buffer l'apparition d'une possibilité de programmer le temps de montée ou *slew rate* afin de choisir :

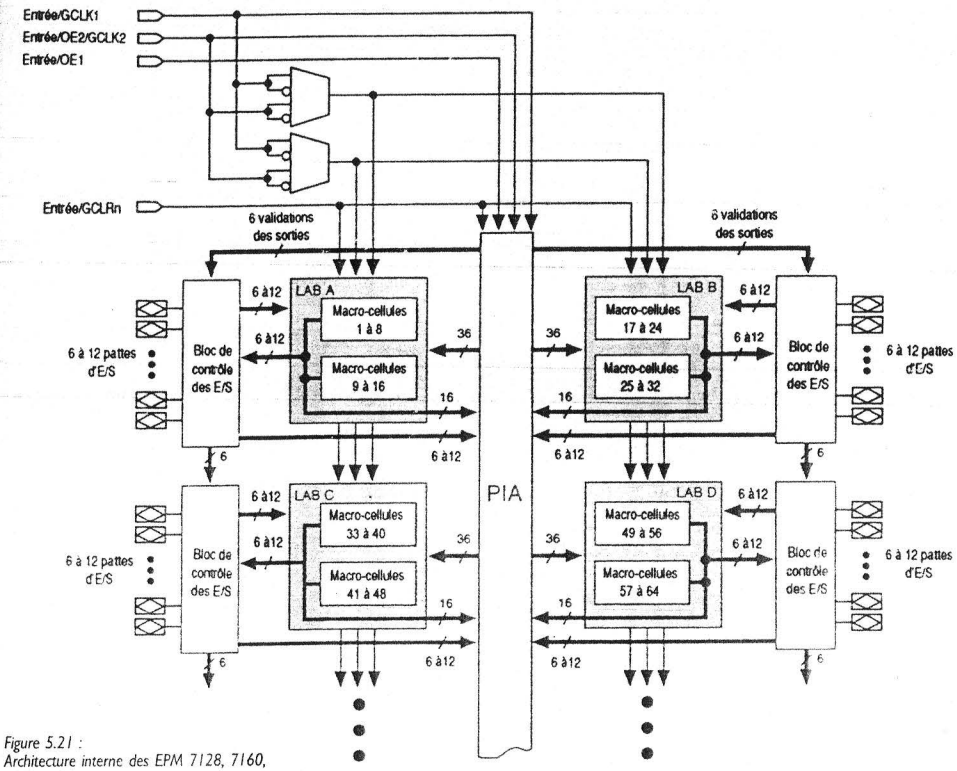


Figure 5.21 :
Architecture interne des EPM 7128, 7160, 7192 et 7256.

– soit une montée "relativement" lente afin de minimiser le bruit généré par la logique ;

– soit une montée très rapide pour les applications où la notion de vitesse est essentielle.

Ce choix de temps de montée est effectué conjointement à la programmation ou non du mode turbo dont nous avons déjà parlé pour les circuits de la série classique, mode qui est évidemment présent ici aussi : les EPLD Altera forment bien une famille cohérente !

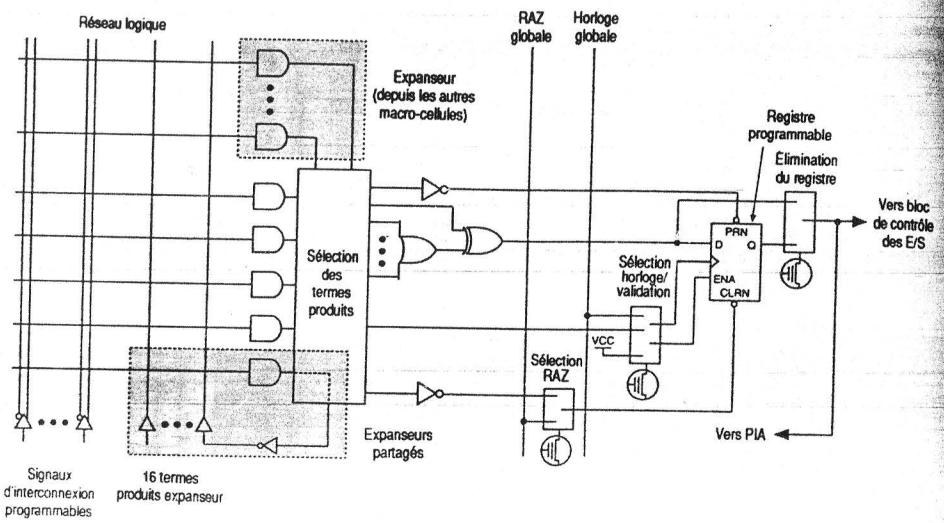


Figure 5.22 :
Schéma de la macro-cellule de la famille MAX 7000.

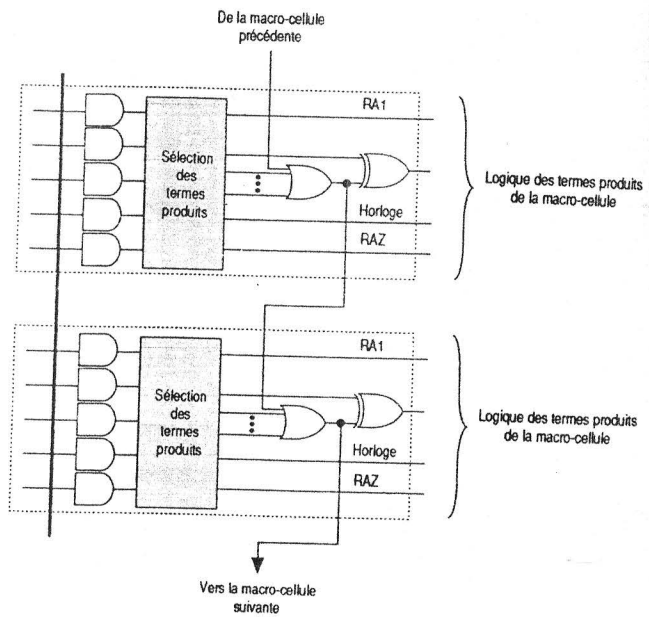


Figure 5.23 :
Partage des ressources entre expanseurs dans la famille MAX 7000.

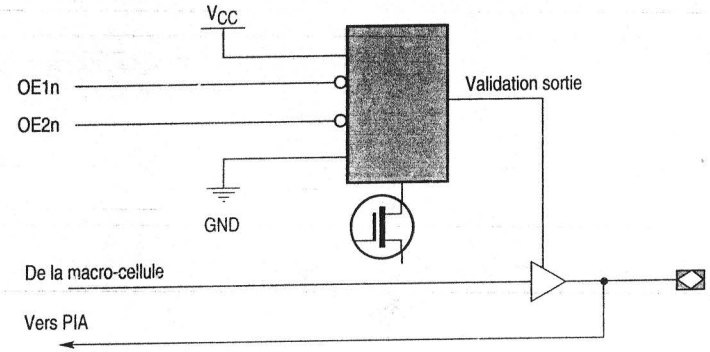


Figure 5.24 :
Contrôle des entrées/sorties des EPM 7032, 7064 et 7096.

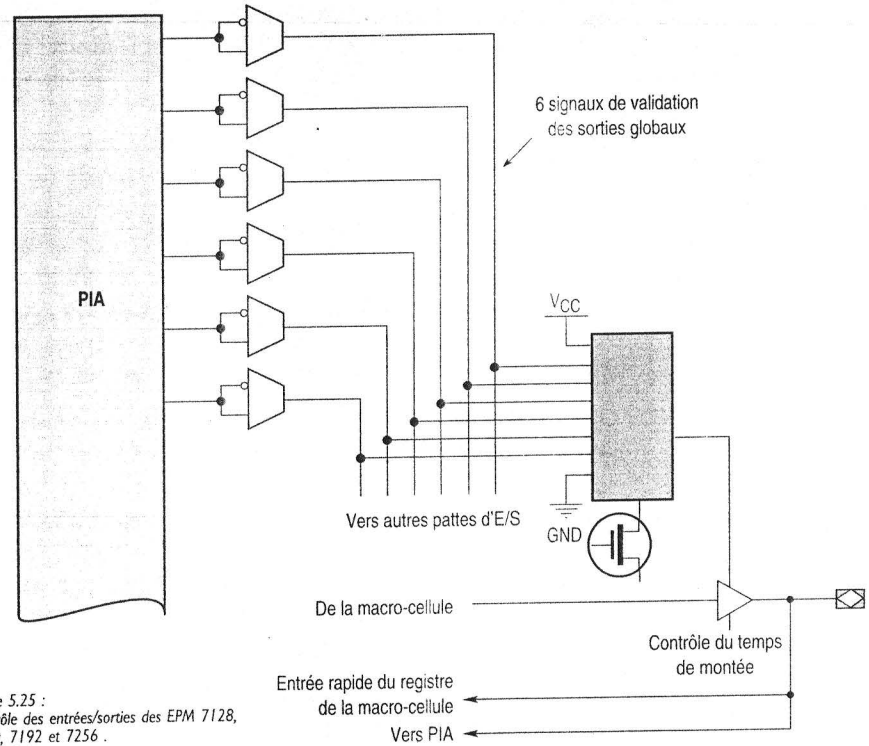


Figure 5.25 :
Contrôle des entrées/sorties des EPM 7128, 7160, 7192 et 7256 .

Contrairement à ce que nous avons fait pour la série MAX 5000, nous ne vous proposons pas de présentation de brochage ou de schéma interne, car cela aurait assez peu d'intérêt dès lors que l'on a vu la figure 5.20 et lorsque l'on sait que le plus gros boîtier de la série MAX 7000 comporte 208 pattes (4 pages de description de son brochage dans sa fiche technique !).

LES EPLD DE LA FAMILLE MAX 9000

Avec ces circuits, nous atteignons une densité d'intégration qui, sans être égale bien sûr à celle des ASIC actuels, est supérieure à celle des ASIC que l'on savait produire il y encore seulement quelques années. Ceci est assez prodigieux lorsque l'on garde présent à l'esprit le fait que ces circuits sont toujours programmables par l'utilisateur avec un logiciel tournant sur un banal compatible PC (en configuration assez *musclée* tout de même mais point n'est besoin de Pentium !).

Les deux innovations majeures concernent la possibilité de programmer les MAX 9000 en circuit et les processus d'interconnexion qui font maintenant appel à la notion de lignes d'interconnexions rapides ou *Fast Track Interconnect*.

La programmation en circuit ne fait appel qu'à une unique tension de 5 volts et respecte les processus normalisés JTAG que l'on rencontre également chez d'autres fabricants (AMD, Lattice, etc.) pour les circuits offrant les même fonctionnalités.

Les lignes d'interconnexions rapides, quant à elles, et comme le montre la figure 5.26, sont des liaisons qui quadrillent le circuit de haut en bas et de droite à gauche entre chaque LAB.

Ces LAB sont toujours constitués de 16 macro-cellules comme dans les circuits MAX précédents, mais ici, chaque LAB reçoit 33 entrées provenant des lignes d'interconnexion rapides les plus proches et 16 signaux de retour provenant du LAB lui même. Tous ces signaux sont disponibles, au sein du LAB, sous forme normale et inversée. En outre, 16 signaux normaux et inversés eux aussi sont disponibles au niveau de l'expanseur, mettant ainsi à disposition de chaque LAB 114 signaux d'entrée utilisables.

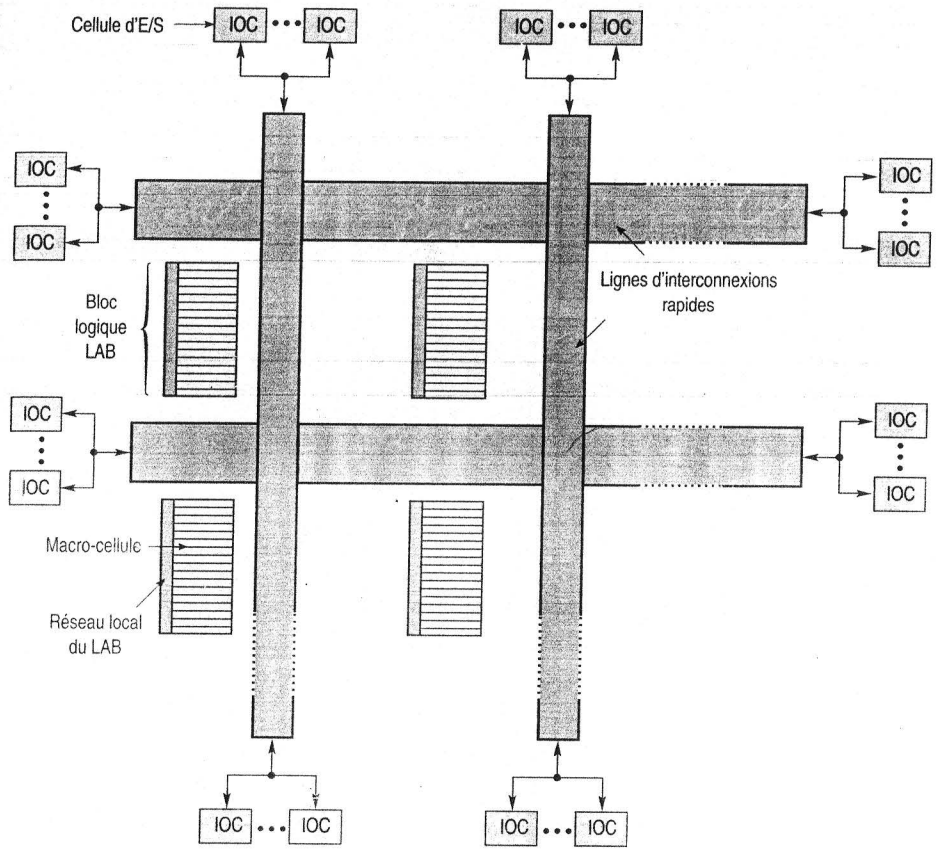


Figure 5.26 : Architecture interne des EPLD de la famille MAX 9000.

En outre, deux lignes rapides d'horloge et de remise à zéro de la bascule des macro-cellules courent de haut en bas de la puce et sont donc également disponibles. La figure 5.27 permet de mieux visualiser cette architecture conjointement à la figure 5.28 qui montre les diverses interconnexions possibles entre les entrées et sorties des LAB et les lignes d'interconnexions rapides horizontales et verticales. Remarquez également qu'il est possible, à ce niveau, d'interconnecter des lignes horizontales et verticales.

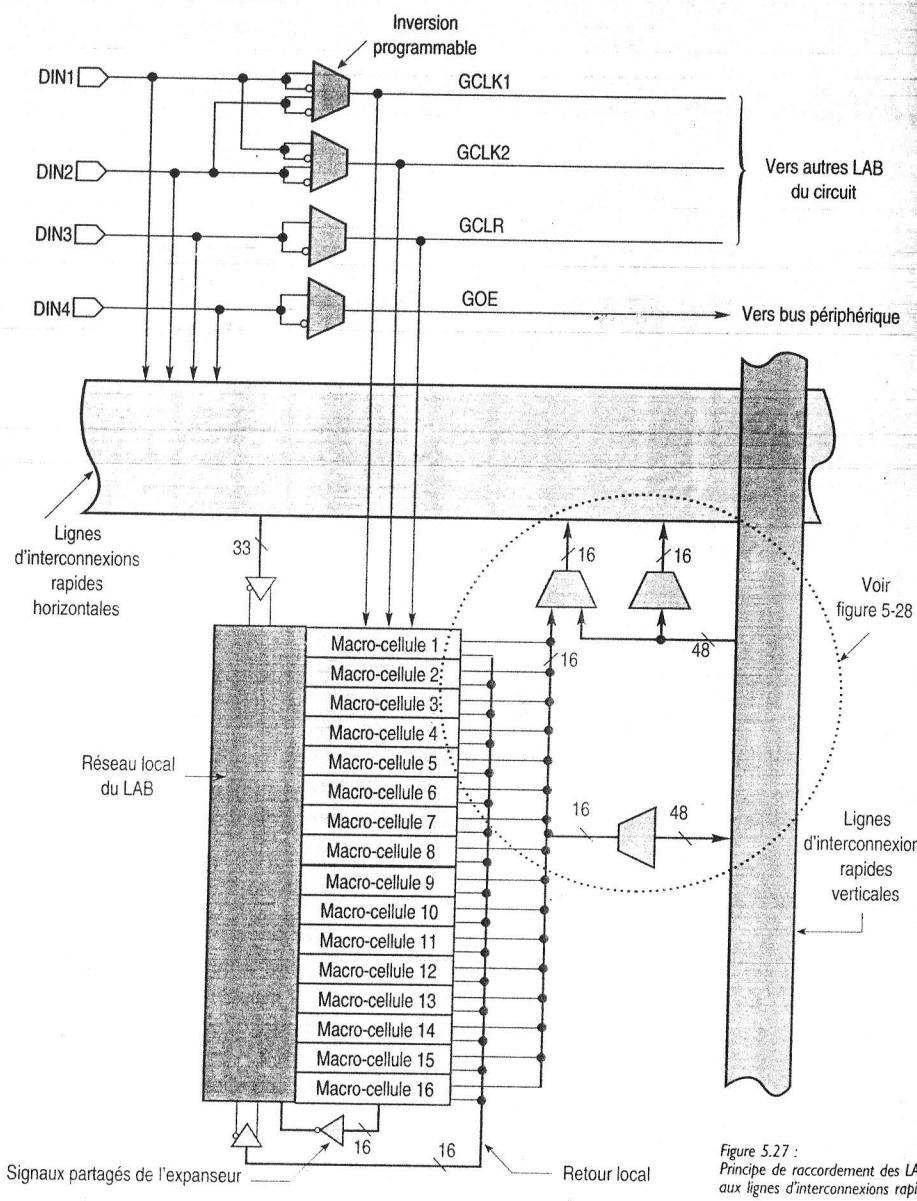


Figure 5.27 :
Principe de raccordement des LAB aux lignes d'interconnexions rapides.

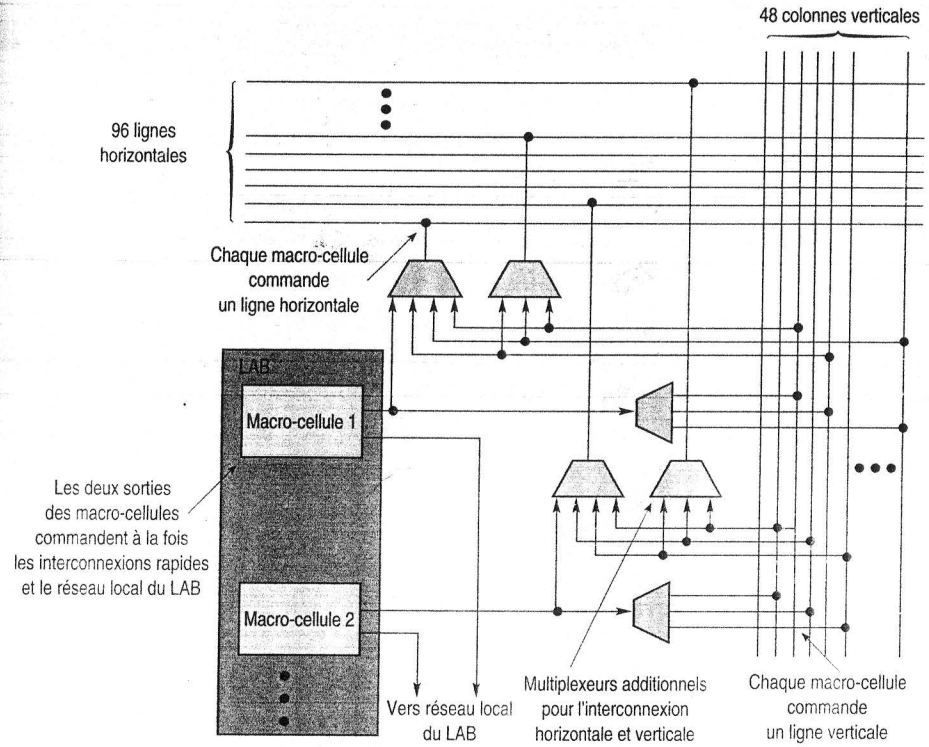


Figure 5.28 :
Détail de la figure 5.27.

Le schéma de la macro-cellule des MAX 9000 reflète bien évidemment ces possibilités comme le montre la figure 5.29. C'est une évolution des macro-cellules des circuits précédents sur laquelle ont été ajoutées les possibilités d'interconnexions supplémentaires.

Les lignes d'interconnexions rapides, qu'elles soient horizontales ou verticales, aboutissent sur les pattes d'entrées/sorties via des IOC ou *Input Output Control* selon les principes visibles figures 5.30 et 5.31. Chaque IOC en mode sortie est piloté par la sortie d'un multiplexeur 1 parmi 10 (lignes horizontales) ou 1 parmi 17 (lignes verticales) sélectionnant ainsi la ligne d'interconnexion rapide de son choix. En outre, chaque IOC en mode entrée peut piloter simultanément 2 lignes d'interconnexions rapides.

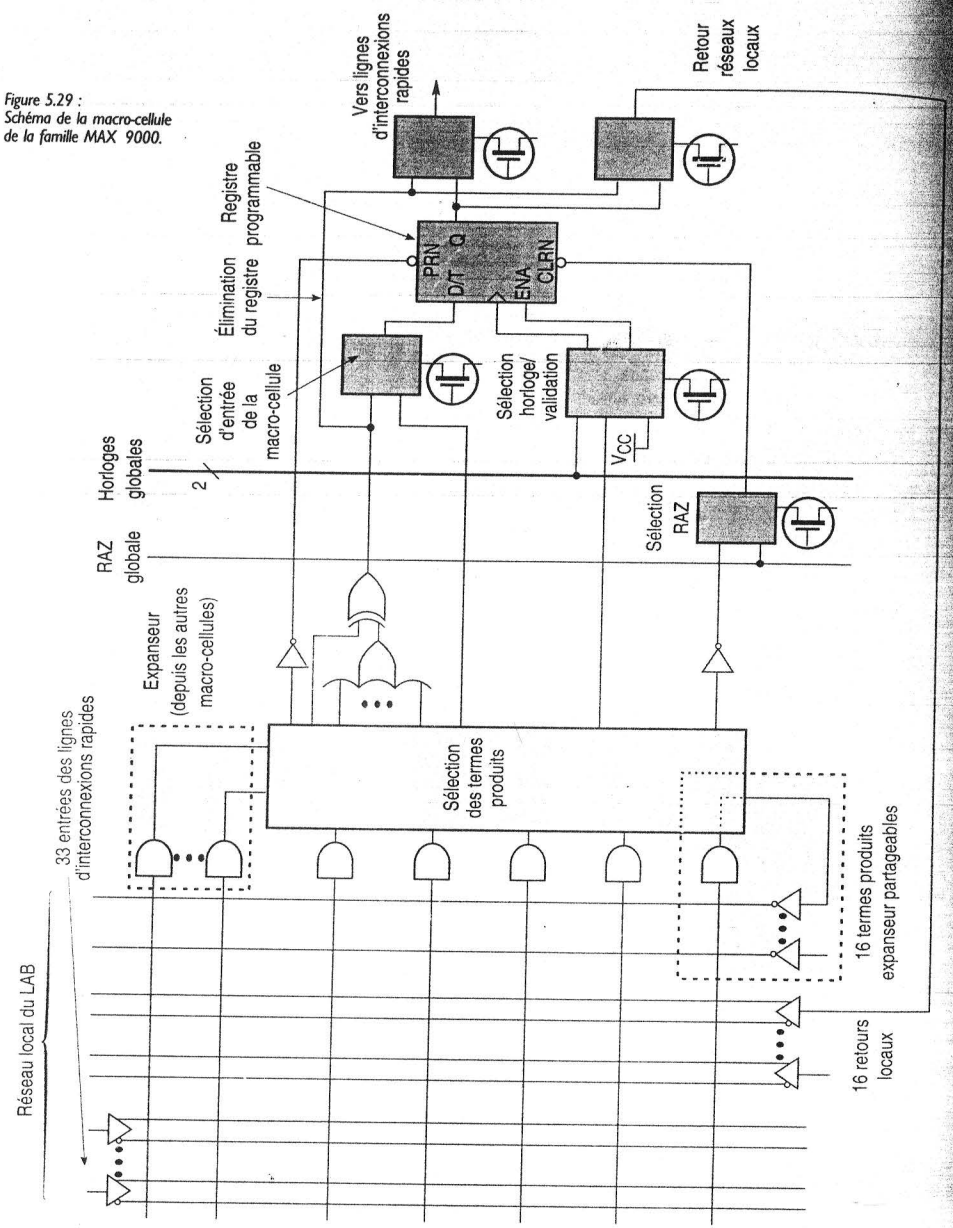


Figure 5.29 :
Schéma de la macro-cellule de la famille MAX 9000.

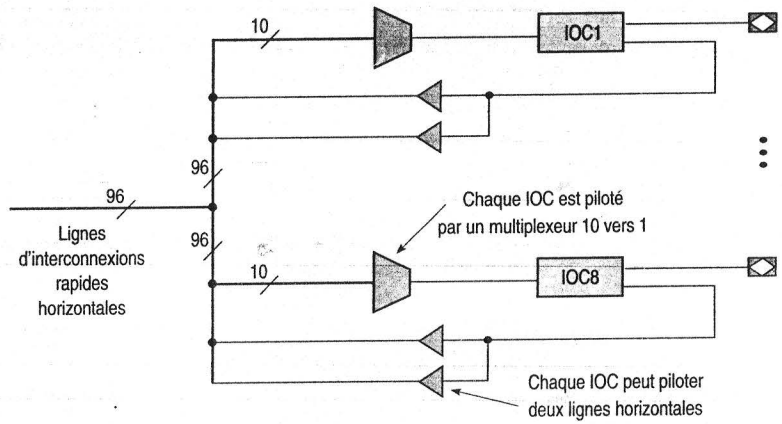


Figure 5.30 :
Raccordement des lignes d'interconnexions rapides horizontales aux IOC.

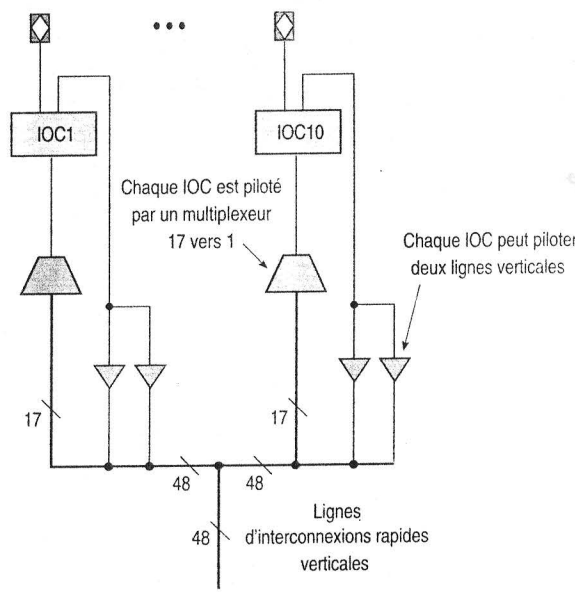


Figure 5.31 :
Raccordement des lignes d'interconnexions rapides verticales aux IOC.

Comme le montre la figure 5.32, chaque IOC peut fonctionner en mode normal ou à registre, les différentes sélections de modes, remise à zéro et horloge du registre pouvant être n'importe quel signal logique provenant du circuit lui-même.

On retrouve bien évidemment, au niveau du buffer de sortie, la possibilité de programmer le temps de montée comme sur la série MAX 7000.

Toujours comme pour la série MAX 7000, nous ne vous proposons pas de présentation de brochage ou de schéma interne car cela aurait assez peu d'intérêt dès lors que l'on a vu la figure 5.26 ,et lorsque l'on sait que le plus gros boîtier de la série MAX 9000 comporte 304 pattes !

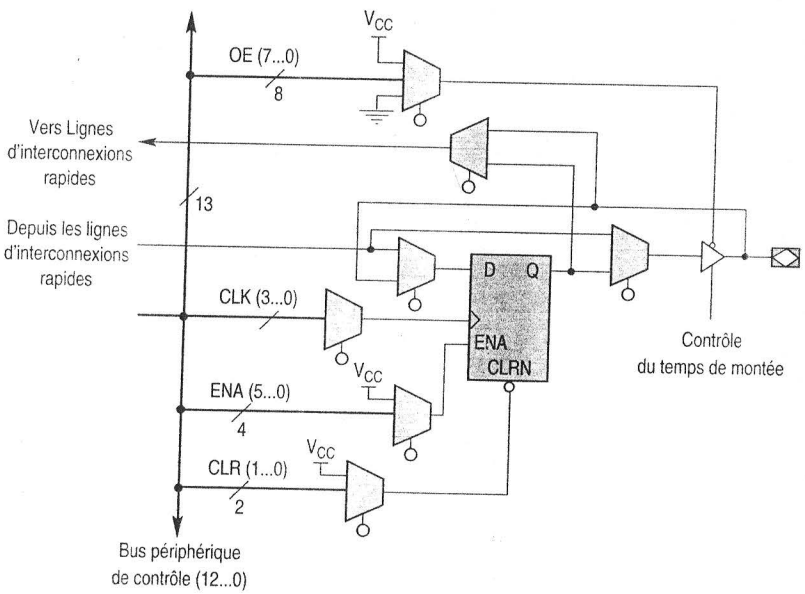


Figure 5.32 : Schéma d'un IOC.

LES EPLD MACH D'AMD

Comme les EPLD MAX 9000 les plus récents, les circuits MACH d'AMD sont des EPLD programmables et effaçables électriquement utilisant de ce fait la technologie E2CMOS. En outre ces composants sont programmables en circuit selon la norme JTAG évoquée ci-avant en n'utilisant qu'une seule et unique tension de 5 volts pour ce qui est des composants placés dans un boîtier comportant plus de 84 pattes.

Deux familles existent actuellement chez AMD, la famille MACH 3 qui est la plus ancienne et ne contient plus qu'un membre et la famille MACH 4 plus récente et plus dense en terme d'intégration comme le montre le tableau résumé de la figure 5.33. En fait, la famille MACH 3 comporte une macro-cellule par entrée/sortie alors que la famille MACH4 en comporte deux.

Circuit	Nombre de pattes	Nombre de macro-cellules	Nombre de portes	Nbre maxi d'entrées	Nbre maxi de sorties	Nbre maxi de bascules	Prog. en circuit	Vitesse (ns)
Famille Mach 3								
Mach 355	144	96	3 500	102	96	96	oui	15,20
Famille Mach 4								
Mach 435	84	128	5 000	70	64	192	non	15,20
Mach 445	100	128	5 000	70	64	192	oui	10,12
Mach 446	100	128	5 000	70	64	192	oui	10,12
Mach 465	208	256	10 000	146	128	384	oui	15,20

Figure 5.33 : Les différents circuits des familles MACH 3 et MACH 4.

Ces circuits adoptant une architecture avec des interconnexions préétablies de longueur fixe, les délais affectant les signaux sont indépendants du routage établi dans le circuit et sont donc parfaitement définis. C'était évidemment le cas aussi de tous les EPLD d'Altera.

LES CELLULES LOGIQUES DE BASE DES MACH

L'approche adoptée par AMD pour ces circuits présente quelques points communs avec les premiers circuits MAX d'Altera puisque, comme le montre la figure 5.34 on retrouve des blocs logiques, qui font un peu penser aux LAB, entourant une zone de connexion centrale que l'on pourrait assimiler à la PIA des MAX 5000.

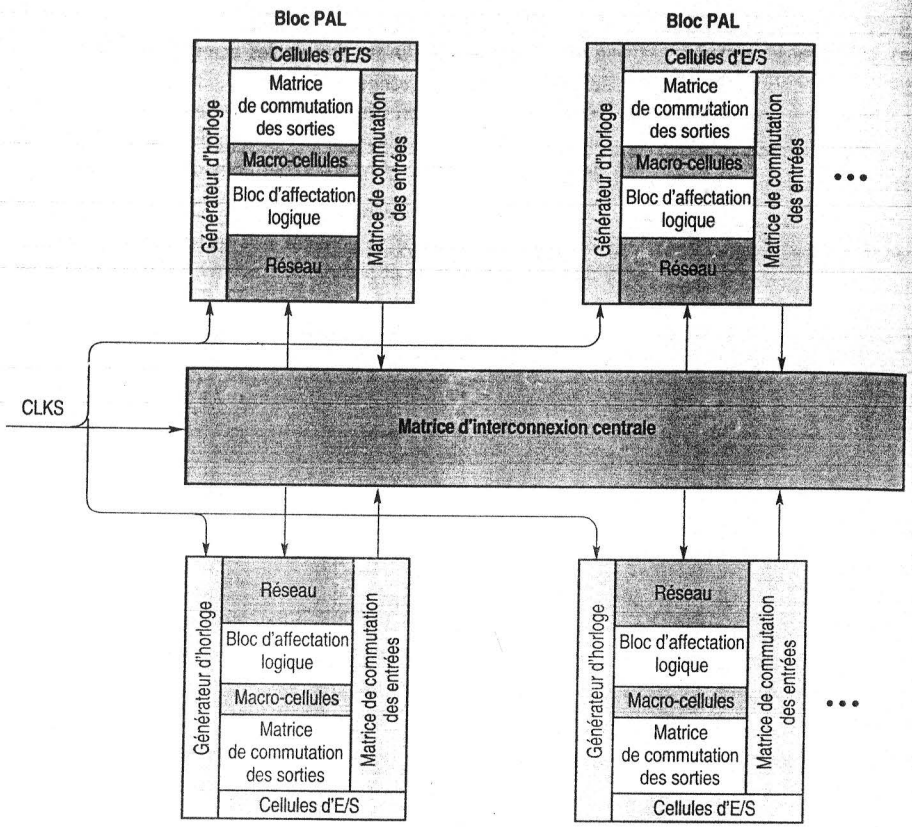


Figure 5.34 :
Architecture interne des circuits MACH.

La similitude s'arrête là, car les possibilités offertes par ces éléments sont différentes. En effet, les bloc logiques équivalents des LAB sont en fait ici analogues à des PAL. Un MACH peut donc être considéré, en première approximation, bien sûr, comme un groupe de PAL (il peut y en avoir jusqu'à 256 dans un même circuit !) interconnectés de façon programmable grâce à la matrice de commutation centrale.

Chaque bloc PAL est constitué de six éléments principaux bien visibles sur la figure 5.34, éléments que nous allons présenter rapidement. On trouve tout d'abord la matrice de commutation d'entrées visible figure 5.35. Les MACH 3 en comptent une par cellule alors que les MACH 4 en disposent de deux.

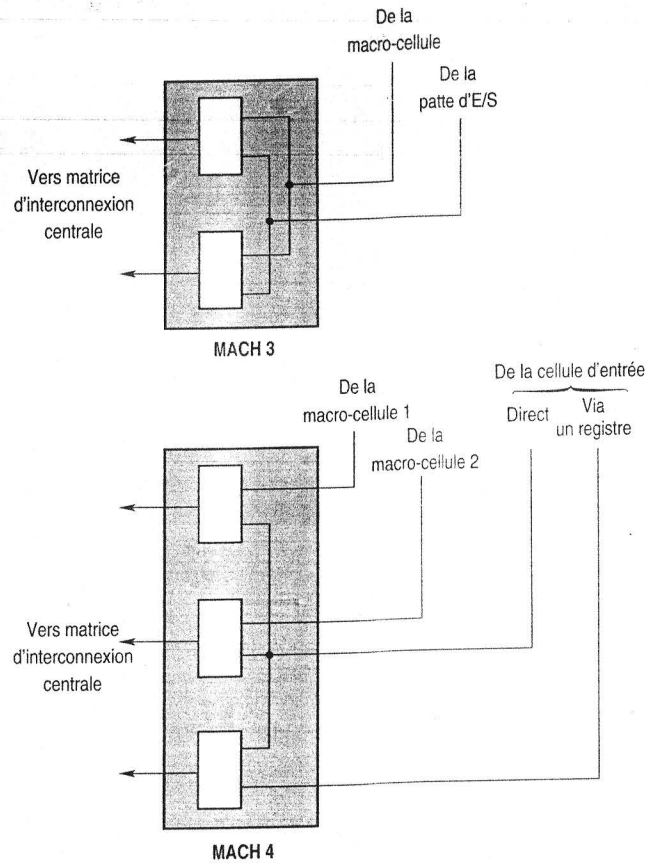


Figure 5.35 :
Schéma de la matrice de commutation d'entrées.

Vient ensuite le bloc d'affectation logique, schématisé figure 5.36. Sa programmation est réalisée automatiquement par le logiciel de placement routage contenu dans l'outil de développement des circuits MACH de façon à optimiser l'utilisation des ressources au sein de chaque bloc PAL.

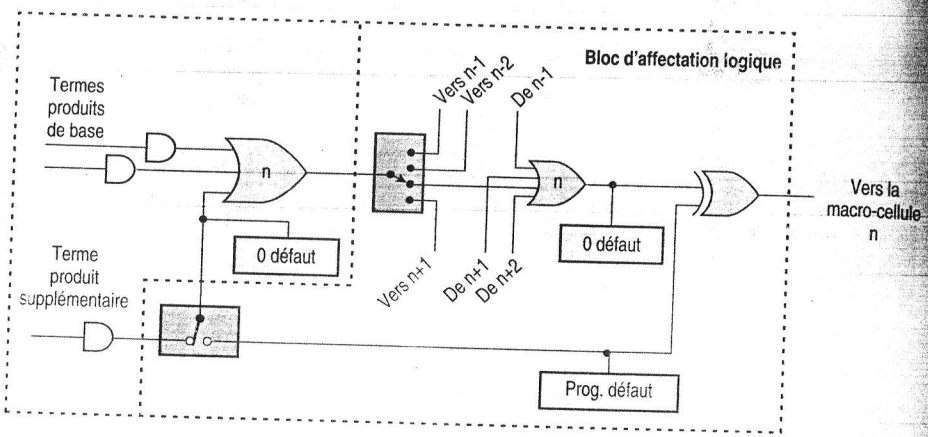


Figure 5.36 :
Schéma du bloc d'affectation logique.

La macro-cellule proprement dite peut sembler relativement simple comme le montre la figure 5.37. Elle peut fonctionner en mode synchrone ou asynchrone et est en fait constituée principalement d'un registre associé à des multiplexeurs permettant de lui faire prendre une des configurations présentées figure 5.38.

Les sorties de cette macro-cellule arrivent sur la matrice de commutation de sorties qui diffère légèrement entre les familles MACH 3 et MACH 4. Les deux configurations vous sont présentées figure 5.39 et 5.40. Dans un MACH 3, chaque macro-cellule peut choisir de commander une cellule d'entrée/sortie parmi 8 et recevoir réciproquement une cellule d'entrée/sortie parmi 8. Le choix de commandes de sortie est réduit à 4 dans la famille MACH 4 tout simplement parce que chaque bloc PAL de cette famille possède deux fois plus de macro-cellules que de cellules de sortie.

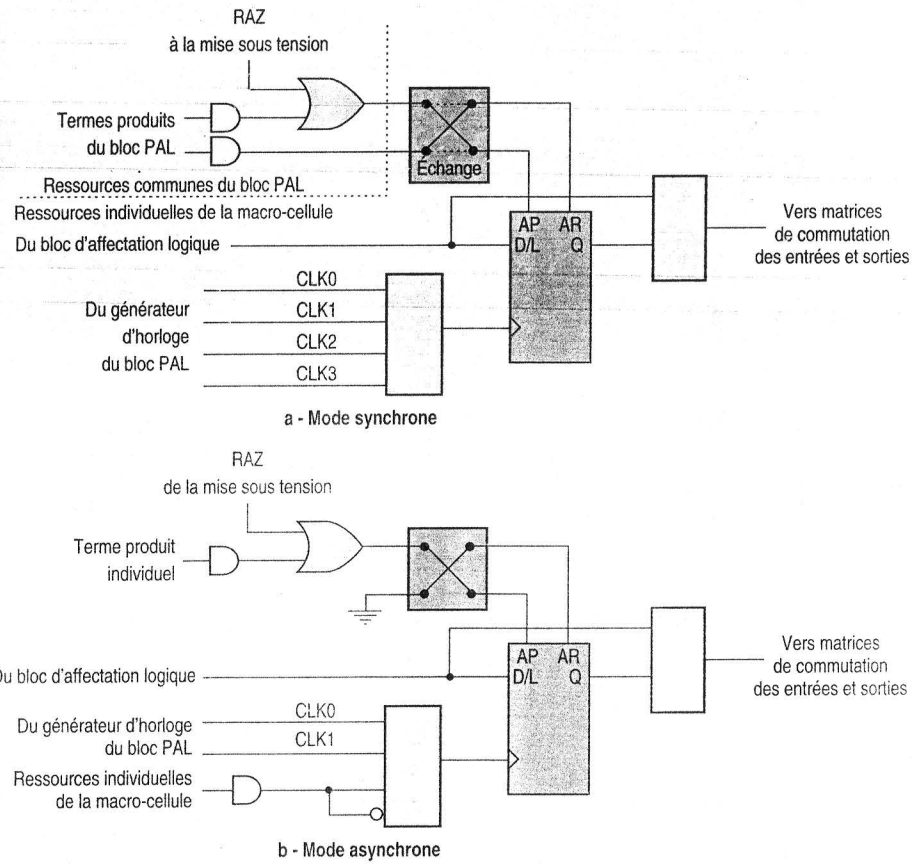
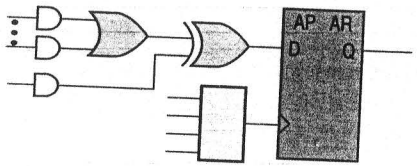
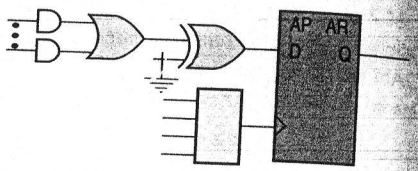


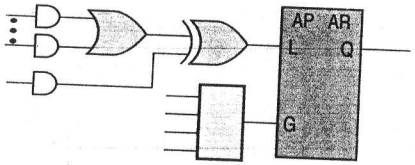
Figure 5.37 :
Schéma de la macro-cellule en mode synchrone et asynchrone.



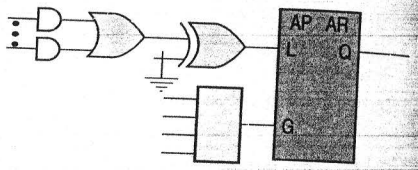
a - Bascule D avec OU exclusif



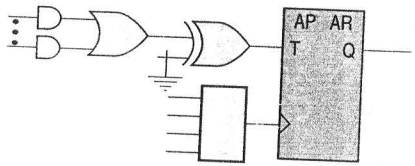
b - Bascule D à polarité du D programmable



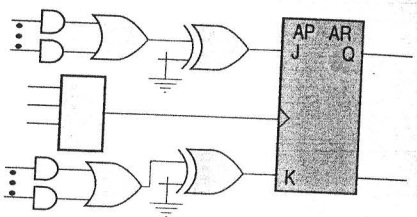
c - Latch avec OU exclusif



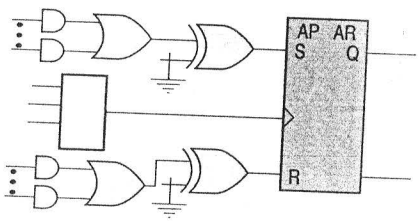
d - Latch à polarité programmable



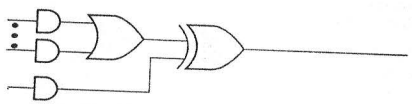
e - Bascule T à polarité du T programmable



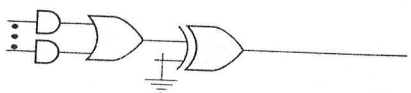
f - Bascule J-K à polarité du J et du K programmables



g - Bascule S-R à polarité du S et du R programmables



h - Combinatoire avec OU exclusif



i - Combinatoire polarité programmable

Figure 5.38 :
Les différentes configurations de la macro-cellule.

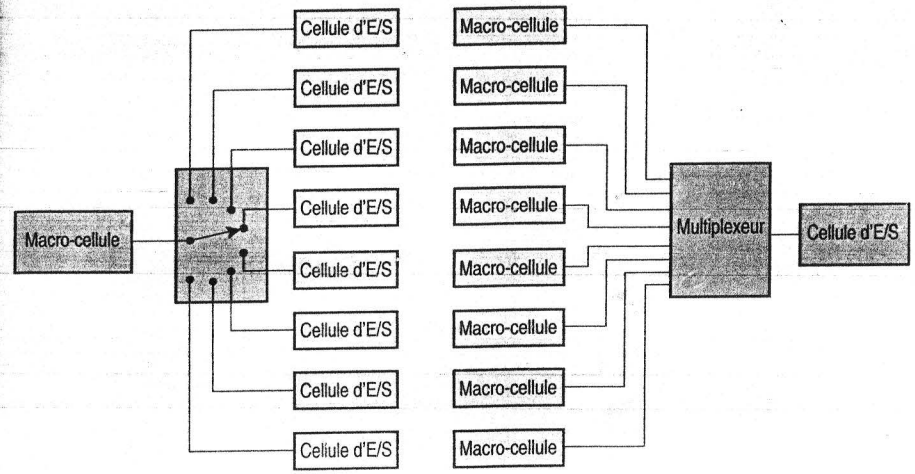


Figure 5.39 :
Schéma de la matrice de commutation des sorties de la famille MACH 3.

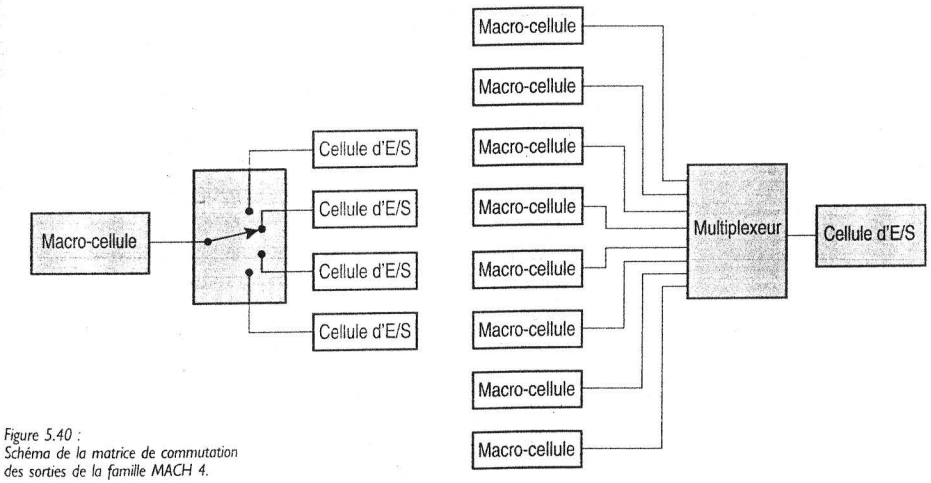


Figure 5.40 :
Schéma de la matrice de commutation des sorties de la famille MACH 4.

Ces cellules d'entrées/sorties justement vous sont présentées figure 5.41. Elles disposent d'un buffer de sortie trois états commandé par la zone programmable et d'un registre ne pouvant toutefois fonctionner qu'en entrée.

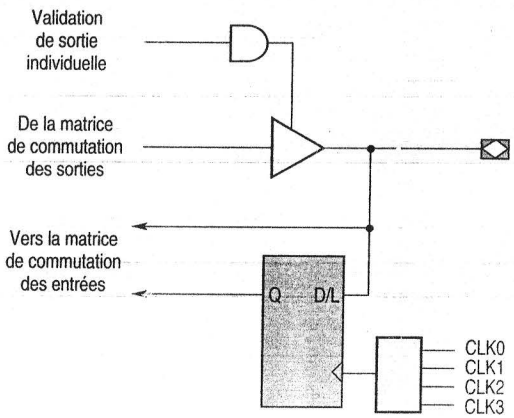


Figure 5.41 :
Schéma des cellules d'entrées/sorties.

Dernière précision qui n'est visible sur aucune des figures : les circuits MACH disposent évidemment d'un bit ou cellule de sécurité interdisant toute lecture de leur contenu une fois qu'il est programmé. Ce bit ne peut être effacé que globalement en même temps que tout le reste du circuit ; la protection apportée est ainsi totale.

LE DÉVELOPPEMENT DES CIRCUITS MACH

De même que le développement des EPLD, quels qu'ils soient, ne pouvait se concevoir qu'avec un logiciel spécialisé, celui des MACH fait de même. AMD propose bien évidemment ses propres outils, mais de nombreux autres fabricants, soit de programmateurs de circuits logiques tel DATA I/O par exemple, soit de logiciels de dessin de schémas et circuits imprimés tel OrCAD, disposent de programmes adaptés. Ces programmes sont évidemment complétés par des logiciels de simulation qui deviennent quasiment indispensables vu la complexité des circuits qu'il est possible de réaliser.

La figure 5.42 donne ainsi une liste, non limitative, de tels fournisseurs à la date de rédaction de cet ouvrage.

Fournisseur	Système de développement	
Advanced Micro Devices, Inc. P.O. Box 3453, MS 1028 Sunnyvale, CA 94088-3543 (800) 222-9323 or (408) 732-2400	MACHXL® Software	MACH355: V1.3 MACH435: V1.3 MACH445: V1.3 MACH465: V1.3
Advanced Micro Devices, Inc. P.O. Box 3453, MS 1028 Sunnyvale, CA 94088-3543 (800) 222-9323 or (408) 732-2400	Design Center/AMD Software	MACH355: V6.1 MACH435: V6.0 MACH445: V6.1 MACH465: V6.1
Advanced Micro Devices, Inc. P.O. Box 3453, MS 1028 Sunnyvale, CA 94088-3543 (800) 222-9323 or (408) 732-2400	AMD-ABEL Software Data I/O MACH Fitters	MACH355: In Development MACH435: V5.0 MACH445: In Development MACH465: In Development
Advanced Micro Devices, Inc. P.O. Box 3453, MS 1028 Sunnyvale, CA 94088-3543 (800) 222-9323 or (408) 732-2400	PROdeveloper/AMD Software PROsynthesis/AMD Software	MACH355: In Development MACH435: In Development MACH445: In Development MACH465: In Development
Cadence Design Systems 555 River Oaks Pkwy San Jose, CA 95134 (408) 943-1234	ComposerPIC™ Designer (Requires MACH Fitter) Verilog, LeapFrog, RapidSim Simulators (Models also available from Logic Modeling)	MACH355: Now MACH435: V3.2 MACH445: V3.3 MACH465: V3.3
Capilano Computing 960 Quayside Dr., Suite 406 New Westminster, B.C. Canada V3M 6G2 (800) 444-9064 or (604) 552-6200	MacABEL™ Software (Requires SmartPart MACH Fitter)	MACH355: Future MACH435: V5.0 MACH445: Future MACH465: Future
CINA, Inc. P.O. Box 4872 Mountain View, CA 94040 (415) 940-1723	SmartCAT Circuit Analyzer	MACH355: Future MACH435: V2.0 MACH445: Future MACH465: Future
Data I/O Corporation 10525 Willows Road N.E. P.O. Box 97046 Redmond, WA 98073-9746 (800) 332-8246 or (206) 881-6444	ABEL™-5 Software (Requires MACH Fitter) Synario™ Software	MACH355: Future MACH435: Now MACH445: In Development MACH465: In Development
iNt GmbH Busenstrasse 6 D-8033 Martinsried, Munich, Germany (89) 857-6667	PLDSim 90	MACH355: Future MACH435: V2.60 MACH445: Future MACH465: Future
ISDATA GmbH Daimlerstr. 51 D7500 Karlsruhe 21 Germany Germany: 0721/75 10 87 U.S.: (510) 531-8553	LOG/iC™ Software (Requires MACH Fitter)	MACH355: In Development MACH435: Now MACH445: In Development MACH465: In Development
Logic Modeling 19500 NW Gibbs Dr. P.O. Box 310 Beaverton, OR 97075 (503) 690-6900	SmartModel® Library	MACH355: Future MACH435: Now MACH445: Now MACH465: In Development
Logical Devices, Inc. 692 S. Military Trail Deerfield Beach, FL 33442 (800) 331-7766 or (305) 428-6868	CUPL™ Software	MACH355: In Development MACH435: V4.4 MACH445: In Development MACH465: In Development

Figure 5.42 :
Quelques fournisseurs de logiciels de développement pour les familles MACH.

LES EPLD pLSI ET ispLSI DE LATTICE

Ces circuits, intermédiaires entre les EPLD et les FPGA, viennent donc logiquement conclure ce chapitre puisque le suivant sera consacré aux FPGA. Comme le montre la figure 5.43, ils existent avec des densités de portes variant de 2000 à 14000, et ont donc une complexité comparable à celle des EPLD d'Altera ou des MACH d'AMD.

Réglons tout de suite le cas des deux appellations : pLSI et ispLSI. En fait, ces deux préfixes correspondent à des circuits aux architectures identiques, mais, alors que les circuits pLSI sont programmables et effaçables électriquement avec un programmateur approprié comme tout bon circuit E2CMOS qui se respecte, les circuits affublés du préfixe isp sont en plus programmables en circuit, conformément au processus JTAG et avec une unique tension de 5 volts. Hormis cette *petite* différence, un pLSI 1032 par exemple est identique à un ispLSI 1032. Nous ne ferons donc plus de distinction dans la suite de ce paragraphe pour ce qui est de la présentation de l'architecture.

L'ARCHITECTURE DES pLSI ET ispLSI

Comme le montre la figure 5.44 et comme vous le constaterez à posteriori à la lecture des deux chapitres suivants, l'architecture des pLSI se rapproche de celle des FPGA où la puce est ceinturée de blocs logiques, appelés ici GLB pour *Generic Logic Bloc*, alors que son centre est une gigantesque zone d'interconnexions appelée ici GRP pour *Global Routing Pool*.

Chaque GLB est constitué de trois éléments fondamentaux comme le montre la figure 5.45 dans le cas des familles pLSI 1000 et 2000. Les 18 entrées du GLB, dont 16 proviennent du GRP et 2 de pattes d'entrées dédiée, arrivent sur un réseau logique capable de générer 20 termes produits. Ces 20 termes produits entrent ensuite dans ce que Lattice appelle PTSA pour *Product Term Sharing Array* afin de générer les 4 signaux de sortie du GLB. En première approximation, on peut considérer le réseau logique d'entrée du GLB comme un réseau de ET programmable, analogue à celui des PAL, et le PTSA comme un réseau de OU programmable. On a donc finalement une structure de type FPLA dans chaque GLB.

Les 4 sorties de cet ensemble attaquent à leur tour 4 OLMC pour *Output Logic Macro Cell*, c'est-à-dire macro-cellule de sortie. Ces macro-

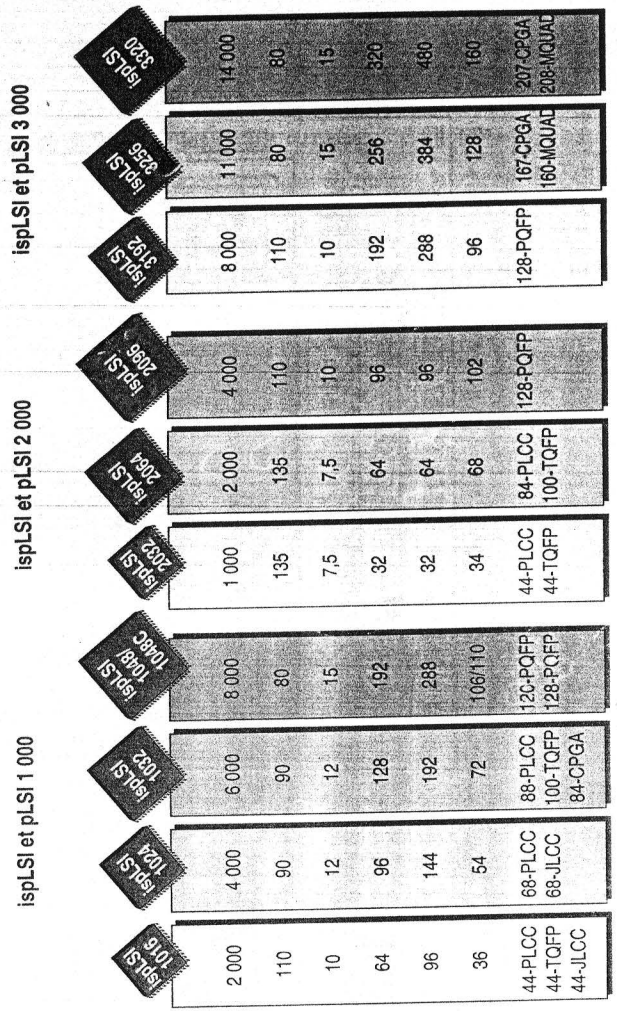


Figure 5.43 :
Les différents circuits pLSI et ispLSI.

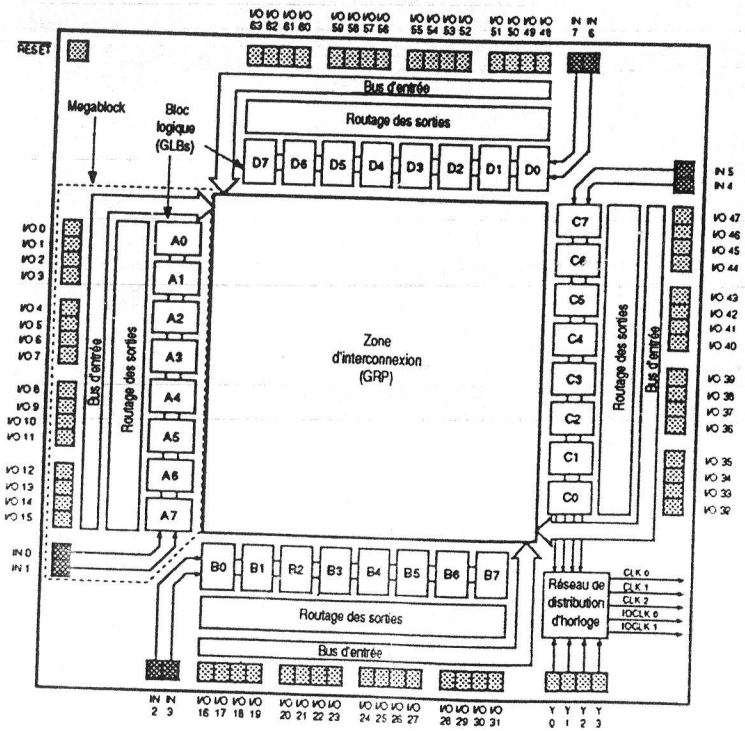


Figure 5.44 :
Architecture interne des circuits pLSI et ispLSI.

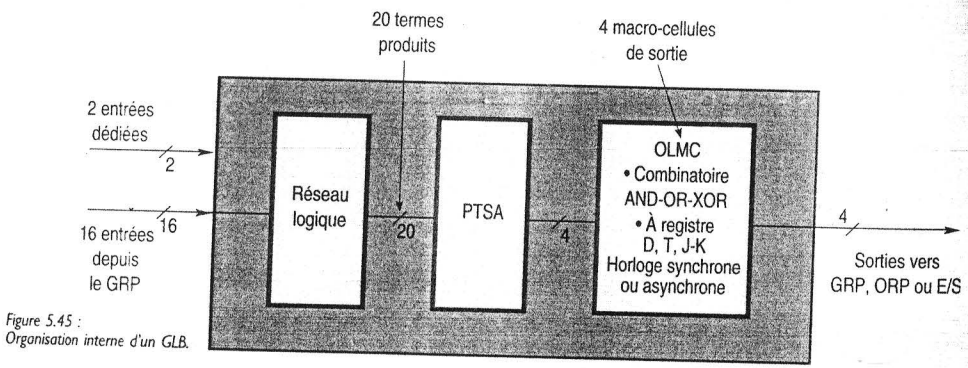


Figure 5.45 :
Organisation interne d'un GLB.

cellules permettent de réaliser des sorties combinatoires ou à registre selon un principe désormais classique.

La figure 5.46 montre de façon plus *électronique* le schéma d'un GLB dans le cas des familles pLSI 1000 et 2000. Pour ce qui est des pLSI 3000, ce schéma est quasiment le même à ceci près que le PTSA et les OLMC sont en double, mais sont toujours alimentés par le même réseau logique d'entrée.

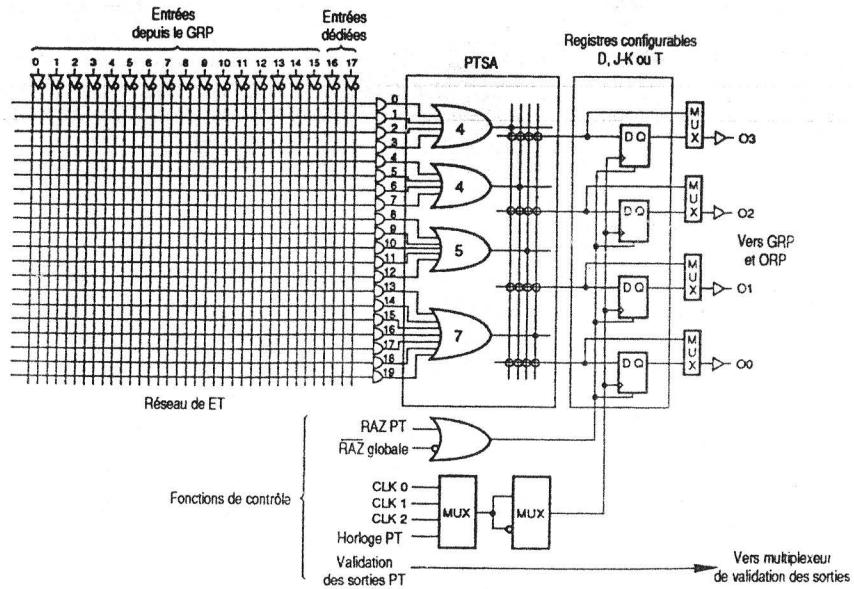


Figure 5.46 :
Schéma d'un GLB.

Comme le laissait deviner la figure 5.44, les GLB sont groupés par 8 dans ce que Lattice appelle un Megablock. Ce groupement contient également 16 cellules d'entrées/sorties ainsi qu'un ORP ou *Output Routing Pool*.

La figure 5.47 montre comment sont traitées les entrées/sorties au niveau de ce Megablock. Tout d'abord, les 16 entrées/sorties du

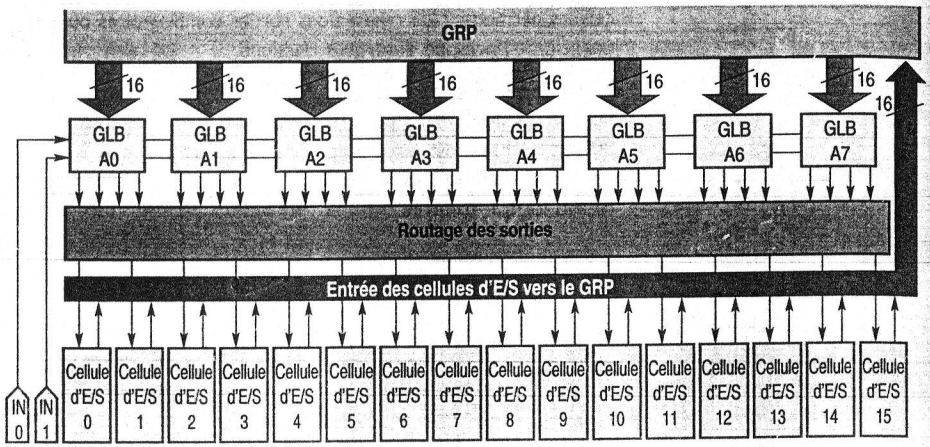


Figure 5.47 :
Organisation interne d'un Megablock.

Megablock sont reliées directement en entrée au GRP, ou bloc d'interconnexions central du circuit, par la liaison visible sur la droite de la figure. Ensuite, chaque Megablock dispose de 2 entrées dédiées directes qui aboutissent sur les GLB sans passer par quoi que ce soit (visibles sur la gauche de la figure).

L'ORP, quant à lui permet de répartir, de la façon la plus souple possible, les sorties des GLB sur les cellules de sorties. Cela se fait au moyen d'un réseau de connexions programmables visible figure 5.48. Le *câblage* de ce réseau est pris intégralement en charge par le logiciel de placement routage contenu dans les outils de développement des circuits pLSI et vous n'avez donc en principe pas à vous en soucier. Sachez enfin qu'existe, au niveau de cet ORP, une possibilité de bypass également schématisée figure 5.48 qui permet d'établir des liaisons préférentielles entre les sorties des GLB et les cellules d'entrées/sorties sans passer par les connexions programmables de l'ORP. Ces liaisons enlèvent bien sûr beaucoup de souplesse à la programmation du circuit et ne doivent donc être utilisées que lorsque les signaux qu'elles véhiculent doivent respecter des timings critiques.

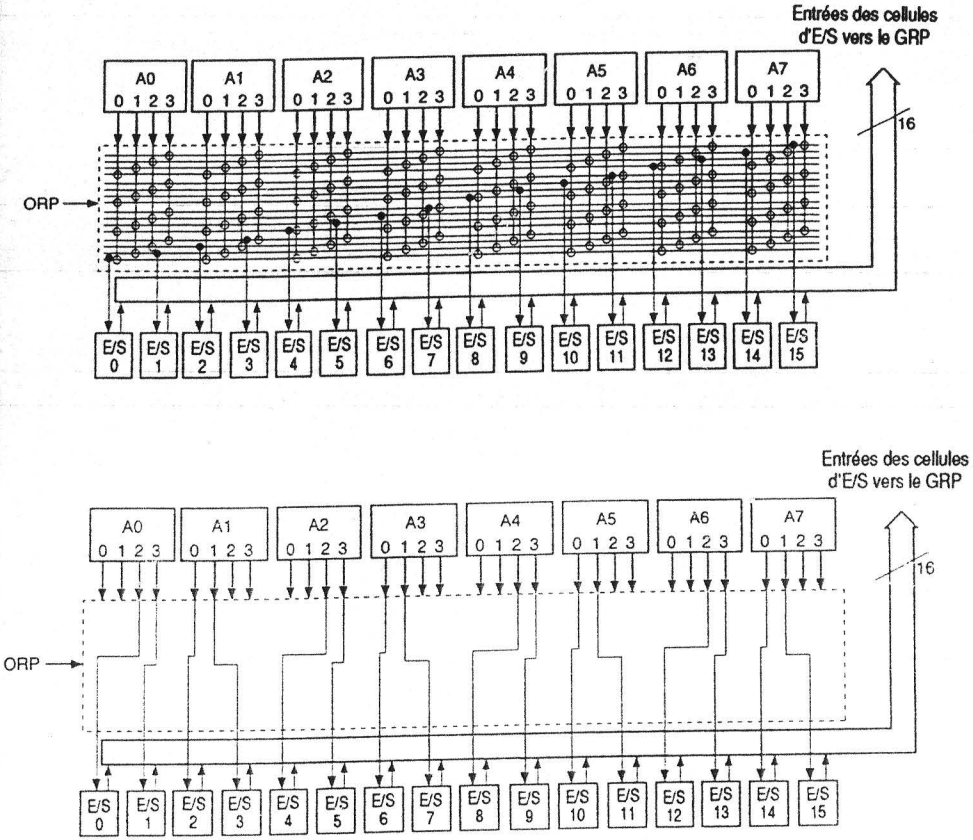


Figure 5.48 :
Principe du routage réalisé dans les ORP.

La cellule d'entrée/sortie est le dernier élément qui nous reste à analyser dans ce schéma. Comme le montre la figure 5.49, elle n'est pas en reste au plan de la souplesse de programmation puisqu'elle permet de réaliser des entrées, sorties, entrées/sorties simples ou à registre comme schématisé figure 5.50.

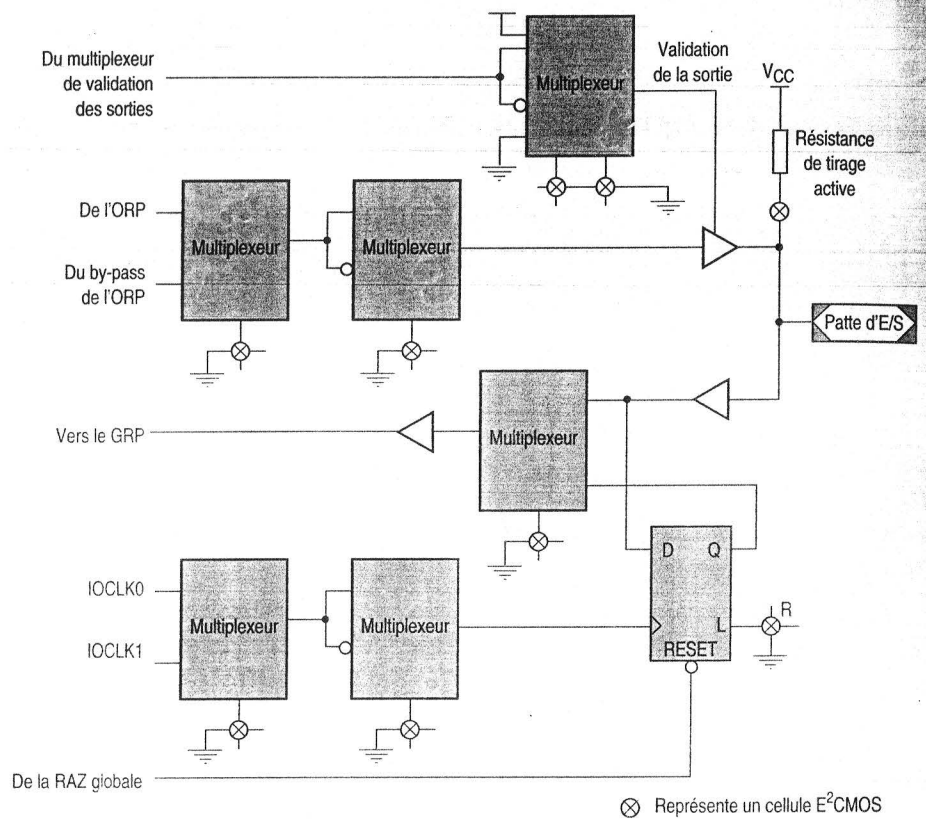


Figure 5.49 :
Schéma de la cellule d'entrée/sortie.

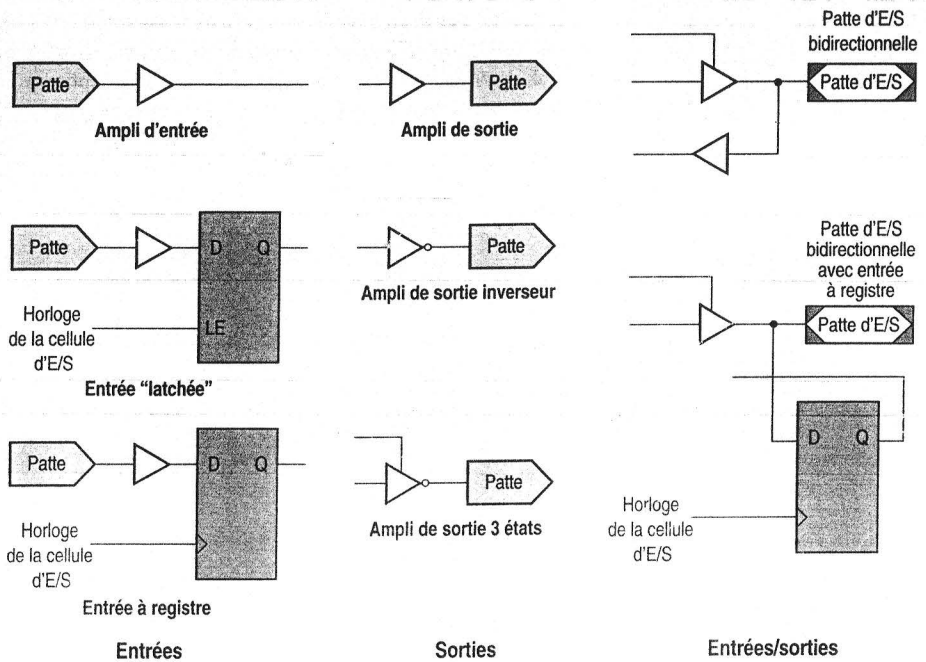


Figure 5.50 :
Les différentes configurations de la cellule d'entrée/sortie.

LE DÉVELOPPEMENT DES CIRCUITS PLSI ET ISPLSI

Lattice propose évidemment ses propres outils de développement, dont un kit de démarrage pour la famille ispLSI fonctionnant sur compatible PC, relativement peu coûteux eu égard aux possibilités offertes. Tous les grands fabricants de programmateurs et de logiciels de dessin de schémas et de circuits imprimés proposent bien sûr eux aussi des logiciels supportant ces familles de circuits.

Les possibilités de ces logiciels sont analogues, dans leurs principes, à celles que nous avons décrites lors de l'étude des EPLD Altera, et nous n'y reviendrons pas ici.

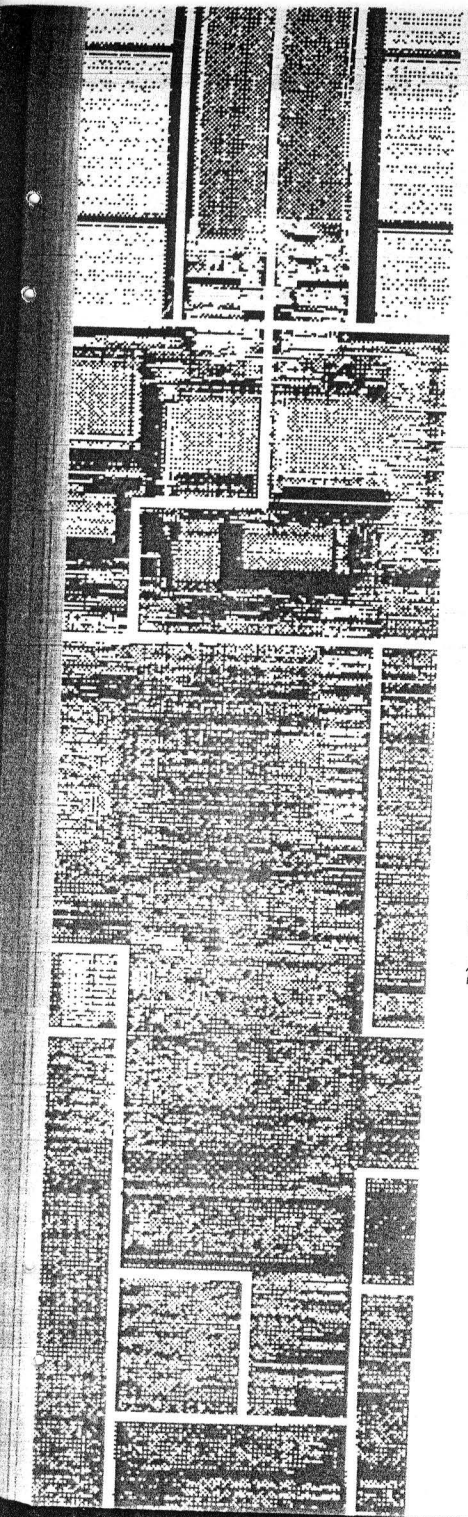
Pour ce qui est des circuits ispLSI, tout ce que nous avons écrit à propos des ispGAL et de leur programmation en circuit au chapitre 4 reste valable, y compris les schémas présentés, tant synoptiques que d'interface avec le port parallèle d'un PC. Le principe de programmation utilisé étant le même pour ces ispLSI que pour les ispGAL, nous vous renvoyons donc à ce chapitre, si bien sûr vous ne l'avez pas déjà lu, pour plus de précision.

CIRCUITS LOGIQUES PROGRAMMABLES

CHAPITRE 6

Les FPGA de type RAM.
FLEX, LCA

- Les LCA de Xilinx 201
- Les Flex d'Altera 232



Nous abordons maintenant une nouvelle famille de circuits logiques programmables par l'utilisateur avec les FPGA. Ce terme, qui signifie* Field Programmable Gate Array*, c'est-à-dire réseau logique programmable sur site, recouvre en fait tous les circuits dont l'architecture s'apparente plus à celle des ASIC qu'à celle des PLD classiques que nous avons vus jusqu'à présent. Cette distinction n'est toutefois pas toujours évidente comme nous l'avons vu en fin de chapitre.

Quoi qu'il en soit, nous allons vous présenter à partir de ce chapitre les circuits appelés FPGA par leurs fabricants et, puisque deux grandes familles technologiques sont en présence, nous allons nous intéresser tout d'abord aux FPGA de type RAM, appelés LCA par Xilinx et FLEX par Altera. Nous verrons ensuite, au chapitre 7, les FPGA à anti-fusibles proposé, entre autres, par Texas Instruments.

LES LCA DE XILINX

Les réseaux logiques programmables de type RAM ou FPGA de type RAM, que nous appellerons LCA dans la suite de cette partie, car c'est plus simple et que c'est aussi leur appellation initiale, ont été inventés par la firme américaine Xilinx et ont été commercialisés pour la première fois en 1984.

Pour concevoir ses produits, Xilinx a étudié les divers types de circuits programmables disponibles sur le marché à l'époque afin de déterminer leurs limitations. L'approche classique des PLD, c'est-à-dire des PAL, GAL et EPLD conduit, selon Xilinx, à un gaspillage relativement important de ressources.

En effet, même si un tel boîtier intègre N portes, un grand nombre d'entre elles sont perdues du fait de leur non utilisation par la matrice d'interconnexion comme le montre la figure 6.1.

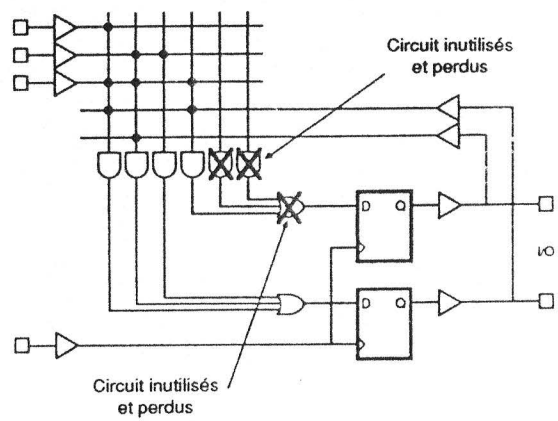


Figure 6.1 :
Mise en évidence du gaspillage de ressources dans un PLD classique.

Ceci tient essentiellement au fait que, si la structure d'entrée des portes *perdues* est bien programmable et offre donc toute la souplesse voulue, la structure de sortie est fixe. Les portes sont connectées à la macro-cellule et un point c'est tout. Si cette dernière n'en a pas besoin, la ou les portes concernées restent donc inutilisées.

La solution idéale, en termes d'utilisation des ressources, est celle proposée par les réseaux de portes programmables par masque qui sont organisés comme le montre la figure 6.2.

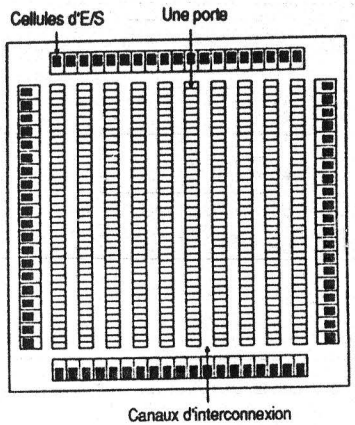


Figure 6.2 :
La structure des réseaux de portes programmables par masque.

En effet, dans cette approche, la puce de silicium est entourée de blocs d'entrées/sorties et son centre est constitué par une matrice de portes élémentaires. Lors de la phase de masquage, ces portes sont interconnectées à la demande étant entendu que sur la puce vierge n'existe aucune connexion préétablie. Le taux d'utilisation des ressources et ainsi excellent mais ce type de circuit n'est programmable que par le fabricant avec tous les inconvénients que l'on sait (coûts, délais, minimum de pièces identiques à commander, outils de développements très lourds à manier et très coûteux).

Xilinx a donc en quelque sorte combiné ces deux approches pour proposer son architecture LCA. Il est à noter dès à présent qu'Altera a procédé à la même analyse quant au gaspillage de ressources des structures de PLD classiques ce qui l'a conduit à proposer ses circuits FLEX, comme nous le verrons en fin de chapitre.

Comme le montre la figure 6.3, l'organisation générale ressemble à celle des réseaux de portes programmables par masque puisque l'on retrouve un ceinturage de la puce par des blocs d'entrées/sorties programmables. Par contre, les portes élémentaires centrales ont été remplacées par ce que Xilinx appelle des CLB pour *Configurable Logic Bloc* que l'on peut assimiler en première approximation aux macrocellules des circuits logiques programmables type EPLD présentés au chapitre précédent.

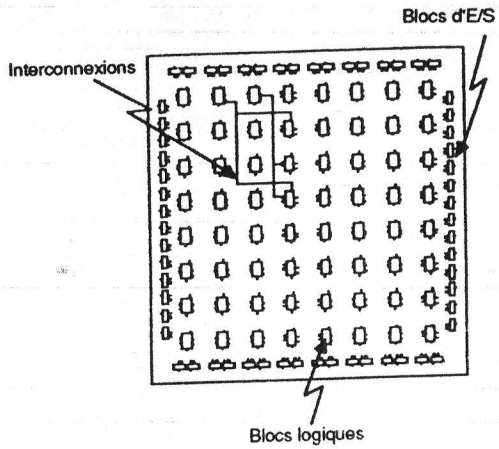


Figure 6.3 :
La structure adoptée par Xilinx pour ses LCA.

Cette utilisation de CLB permet de réduire le nombre d'interconnexions nécessaires entre les différents blocs par rapport aux portes élémentaires que l'on trouve dans les réseaux programmables par masque. Mais le point fort des LCA ne se limite pas à cela ; en effet, nous n'avons pas encore expliqué comment était résolu ce fameux problème d'interconnexion et c'est cela qui fait toute l'originalité et toute la souplesse d'utilisation des LCA.

En fait, il existe dans tout LCA des matrices de lignes d'interconnexions qui courent horizontalement et verticalement entre les divers CLB. Les connexions sur ces lignes sont effectuées par des transistors MOS dont l'état est contrôlé par des cellules de mémoire vive ou RAM.

Comme le montre schématiquement la figure 6.4 on peut donc considérer que sous le réseau logique proprement dit se trouve une zone de mémoire dans laquelle est chargée la configuration de connexions à établir.

Une RAM étant une mémoire dans laquelle on peut lire ou écrire autant de fois que l'on veut et sans aucune restriction ; on conçoit que les LCA soient ainsi programmables et effaçables électriquement bien sûr, mais, comme ces opérations sont en fait de simples écritures mémoire, elles ne prennent que quelques millisecondes et ne font

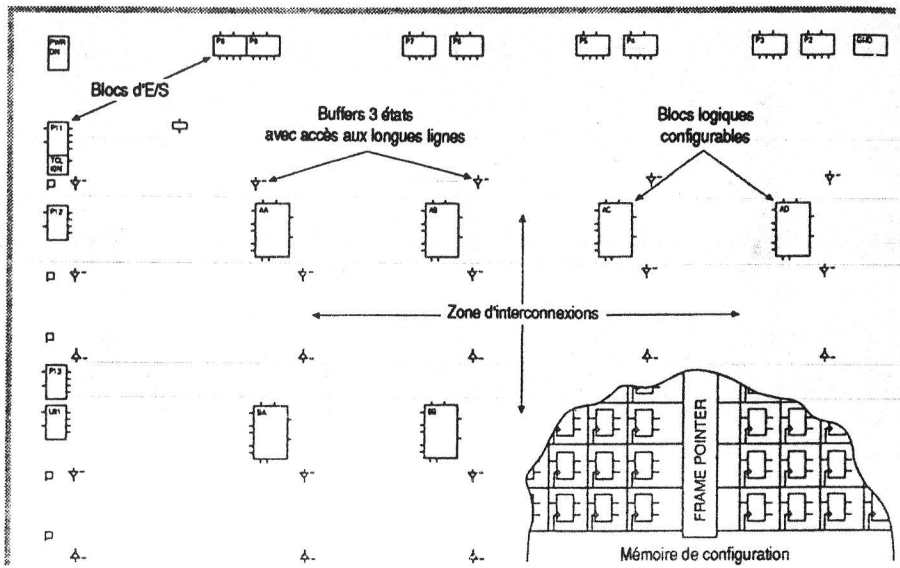


Figure 6.4 :
La configuration des connexions des LCA fait appel à une RAM enterrée sous la logique.

appel à aucun programmateur spécialisé. Elles n'utilisent en effet aucun haute tension ou tension spéciale et se contentent de l'alimentation 5 volts du circuit lui même.

Les LCA étaient donc les premiers composants logiques programmables en circuit et, si l'on se remémore ce qui existait en 1984 où quasiment personne n'envisageait cette possibilité, cela représentait une véritable petite révolution. Seul petit bémol, cette possibilité n'était pas due à une technologie E2CMOS, pas encore au point pour ce faire à l'époque, mais à la présence de RAM statique dans le LCA ce qui induisait quelques petites contraintes.

En effet, comme vous le savez certainement, une RAM perd son contenu lors de toute coupure d'alimentation. Notre LCA, si l'on ne prend pas de précautions particulières, va donc oublier sa configuration de connexions lors de chaque arrêt de l'appareil dans lequel il se trouve. Pour remédier à cela trois solutions sont envisageables :

- L'utilisation d'un support spécial intégrant une petite pile au lithium qui maintient le LCA alimenté même en cas de coupure secteur. Comme la consommation du plan mémoire est dérisoire au repos, la durée de vie de la pile peut être de l'ordre de 3 ans.
- L'utilisation d'une mémoire morte, de type PROM bien sûr qui, lors de chaque mise sous tension, se recopie en quelques millisecondes dans le LCA et le reconfigure ainsi immédiatement. Cette façon de faire est évidemment nettement préférable à la précédente, car il n'y a plus à craindre de perte d'information suite à l'épuisement d'une quelconque pile.
- L'utilisation des ressources d'un microprocesseur ou microcontrôleur se trouvant dans le même appareil que le LCA et qui peut alors télécharger une partie de sa mémoire dans le LCA pour le configurer lors de chaque mise sous tension.

On le voit, du fait de cette utilisation d'une RAM pour configurer les connexions, le LCA est un peu contraignant pour ce qui est de la phase de mise en marche. Par contre, il ouvre des horizons nouveaux aux concepteurs de circuits car, du fait de cette RAM qui peut être lue mais surtout écrite à tout instant, il est possible de modifier la configuration de la logique réalisée par le circuit alors même que le montage qui l'utilise est en fonctionnement.

LES CELLULES LOGIQUES DE BASE DES LCA

Comme nous l'avons laissé entendre précédemment, les LCA utilisent deux types de cellules de base :

– les cellules d'entrées/sorties, appelées ici IOB pour *Input Output Bloc* ;

– les cellules logiques proprement dites appelées CLB pour *Configurable Logic Bloc*.

Voyons tout d'abord ce qu'il y a à l'intérieur d'un IOB grâce à la figure 6.5 qui présente de façon aussi schématique que possible cette structure qui peut être entrée, sortie ou entrée/sortie.

Nous voyons tout d'abord en partie haute cinq rectangles qui ne correspondent pas à des connexions au sens propre du terme, mais à des sélections de modes de fonctionnement. Ces sélections sont cepen-

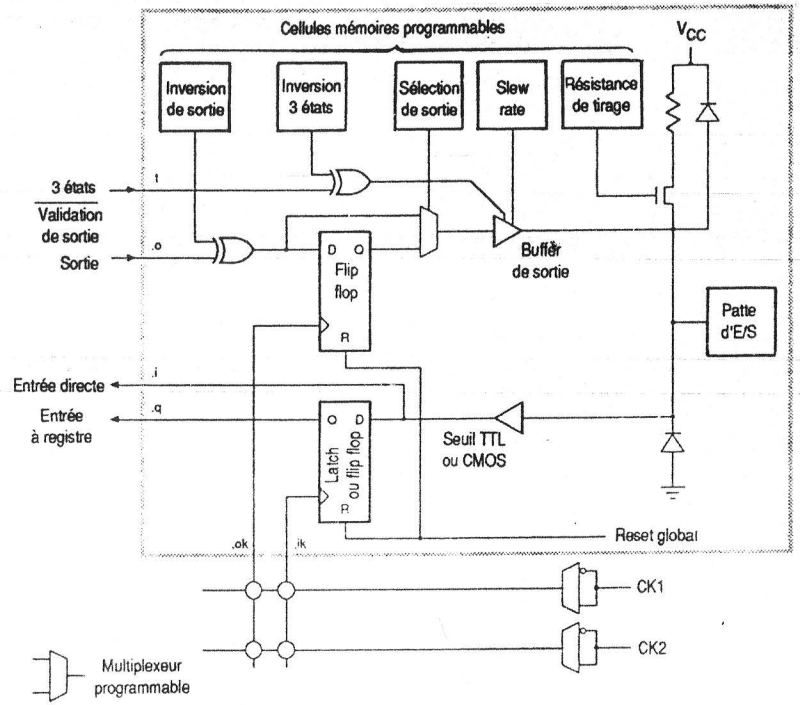


Figure 6.5 :
Structure détaillée d'un IOB.

dant programmables et configurables avec la même souplesse que les connexions et font bien évidemment appel aux cellules de RAM que nous venons d'évoquer. On dispose des possibilités suivantes :

- Inversion ou non du signal avant son application à l'IOB.
- Inversion ou non du signal de commande du buffer de sortie qui est du type trois états.
- Sélection du type de sortie avec ou sans intervention du registre situé en partie haute de la figure 6.5.
- Sélection du slew rate ou temps de montée du signal de sortie pour s'adapter à la logique connectée au LCA ;

- Mise en place d'un *pull-up* passif ou résistance de tirage au niveau haut lorsque la patte est en entrée (ce qui permet de la laisser en l'air tout en fixant son niveau comme dans tout bon circuit intégré qui se respecte).

Ceci étant vu, le signal de sortie peut passer ou non par une bascule de type D. Cette dernière est remise à zéro par un signal de reset global qui est activé automatiquement lors de la phase de configuration du circuit permettant ainsi un démarrage dans un état parfaitement défini. Ce reset peut également être activé par une patte dédiée du boîtier.

L'horloge de cette bascule est librement commandée par le signal de votre choix en provenance de la zone d'interconnexions dont nous parlerons dans la suite de cette présentation. Le changement d'état peut avoir lieu sur un front montant ou descendant au choix, et la bascule peut également fonctionner comme un classique latch transparent.

Le signal d'entrée, quant à lui, peut passer ou non au travers d'une bascule du même type qui dispose des mêmes possibilités de connexions et de travail que la bascule de sortie.

Comme vous pouvez le constater, ce bloc d'entrées/sorties ou IOB dispose donc de toutes les fonctionnalités souhaitables quelle que soit l'application à laquelle on le destine.

La cellule logique ou CLB n'est pas en reste comme vous pouvez déjà le vérifier à l'examen de sa structure présentée figure 6.6. Elle est composée d'un certain nombre de sous-ensembles de base qui sont :

- deux bascules de type D ;
- un bloc de logique combinatoire ;
- divers multiplexeurs utilisés comme commutateurs pour sélectionner les divers modes de fonctionnement de la cellule.

Voyons tout d'abord le bloc logique car c'est cet élément qui pose en général le plus de problèmes (à tort) à l'utilisateur novice d'un LCA. Il dispose de cinq entrées logiques directes *a* à e, de deux entrées qui proviennent des bascules QX et QY et de deux sorties F et G et peut

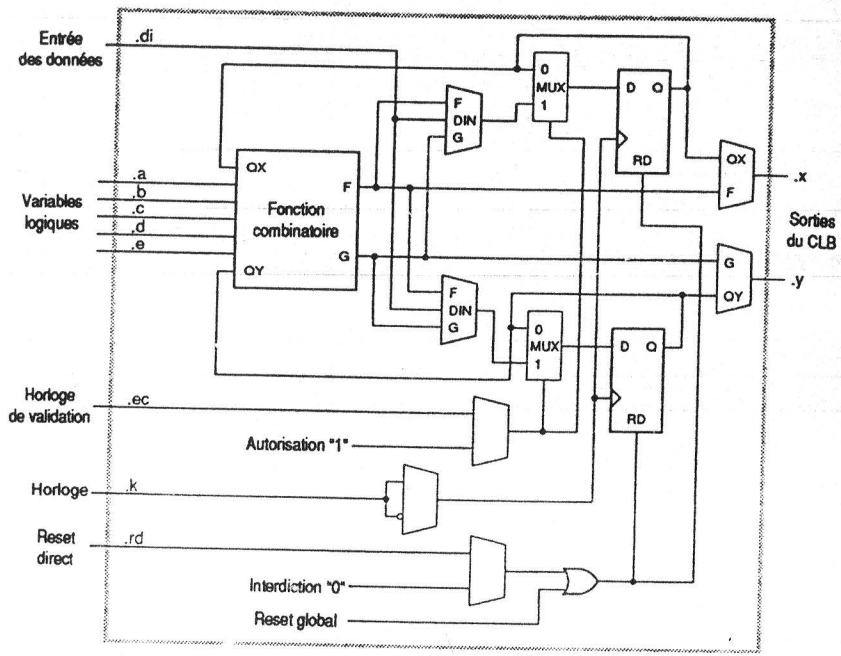


Figure 6.6 :
Structure détaillée d'un CLB.

réaliser n'importe quelle fonction logique combinatoire de ces entrées. Pour préciser cela examinons la figure 6.7 qui présente les différents choix possibles. En fait, ce bloc peut travailler dans trois modes distincts : le mode FG, le mode F et le mode FGM.

Dans le mode FG présenté en partie haute de la figure 6.7, il peut générer deux fonctions logiques combinatoires indépendantes de quatre variables chacune. Pour ce faire, la variable A doit être commune aux deux fonctions, les deuxième et troisième variables peuvent être choisies librement parmi B, C, QX et QY et la quatrième variable peut être D ou E.

Le deuxième mode de fonctionnement ou mode FG permet de réaliser n'importe quelle fonction logique de cinq variables comme le montre la partie centrale de la figure 6.7. Les variables sont nécessairement A, D et E et deux autres choisies parmi B, C, QX et QY.

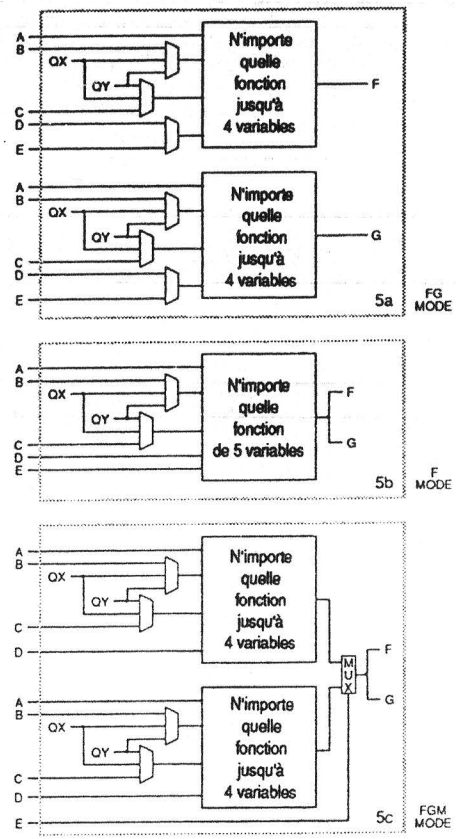


Figure 6.7 :
Les différents modes de fonctionnement du bloc logique d'entrée du CLB.

Enfin, le dernier mode ou mode FGM permet d'utiliser E comme sélecteur de sortie pour deux fonctions de quatre variables. Ces fonctions utilisent nécessairement comme entrées A et D et deux autres variables à choisir parmi B, C, QX et QY.

On le voit, ce bloc logique est d'une très grande souplesse d'emploi et ce, d'autant que si vous utilisez un logiciel de développement avec la fonction APR que nous verrons ensuite, vous n'avez même pas à vous soucier de ces sélections de modes que le logiciel fera automatiquement.

Ceci étant vu, les sorties de ce bloc logique peuvent être appliquées à des bascules ou directement à la sortie du CLB. Les bascules disposent d'une entrée de reset activée par le signal de reset global dont nous venons de parler pour les IOB. Cette entrée peut également être commandée par un signal provenant de la zone d'interconnexions.

L'horloge des bascules et le signal de validation de cette horloge proviennent aussi de la zone d'interconnexions et la polarité d'action de cette horloge (front montant ou descendant) peut être librement choisie par programmation.

Si vous ajoutez à toutes ces possibilités de configuration et de programmation le fait qu'un circuit tel que le XC 3090 par exemple ne contient pas moins de 320 CLB et 144 IOB, vous avez déjà une certaine idée des possibilités d'intégration considérables offertes par les LCA.

LES DIFFÉRENTS TYPES D'INTERCONNEXIONS

Il ne suffit pas de disposer de blocs logiques ou d'entrées/sorties performants, encore faut-il pouvoir les interconnecter avec un maximum d'efficacité afin que leur taux d'utilisation dans un circuit donné soit aussi élevé que possible.

Pour y parvenir, Xilinx propose trois types de moyens d'interconnexions différents que nous allons décrire maintenant. Bien sûr, tous reposent sur l'établissement ou non de liaisons avec des transistors MOS pilotés par de la mémoire RAM, puisque c'est le concept de base des LCA mais, selon la longueur et la destination des liaisons, les cheminements utilisés ne sont pas les mêmes ; on distingue en effet :

- Les interconnexions à usage général.
- Les interconnexions directes.
- Les lignes longues ou long lines.

Tous ces chemins d'interconnexions sont accessibles au niveau des transistors MOS de commutation matérialisés par ce que Xilinx appelle les PIP ou Programmable Interconnect Points repartis sur toute la puce comme le montre un extrait du dessin de celle-ci visible figure 6.8.

Les interconnexions à usage général consistent en une grille de cinq segments métalliques verticaux et de cinq segments métalliques hori-

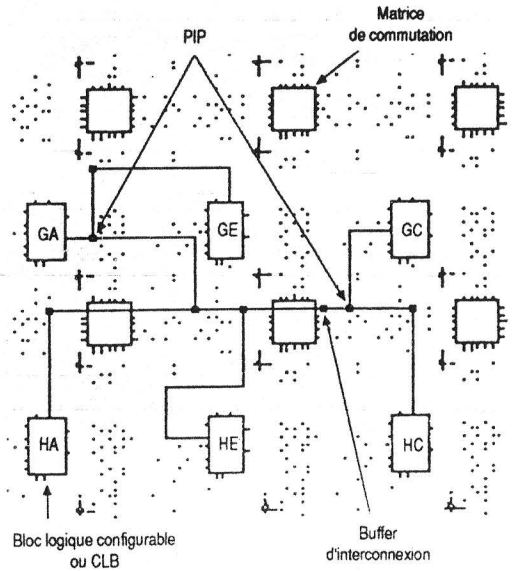


Figure 6.8 : Mise en évidence de quelques PIP.

zontaux placés entre les rangées et les colonnes de CLB et d'IOB. Chaque segment fait la hauteur ou la largeur d'un bloc logique. À chaque intersection de rangée et de colonne se trouve placée une matrice de commutation qui permet de raccorder les segments entre eux selon diverses configurations présentées figure 6.9.

Même si toutes les combinaisons ne sont pas autorisées, le nombre de variantes disponibles permet déjà une grande liberté.

Afin que les signaux qui parcourent de grandes distances via ces lignes sur la puce ne soient pas trop affaiblis, des buffers sont implantés régulièrement en haut et à droite de toutes les matrices de commutation. Il est alors possible d'y faire passer les signaux qui le nécessitent.

Les interconnexions directes, dont le principe est visible figure 6.10, permettent l'établissement de liaisons entre les CLB et les IOB avec un maximum d'efficacité en termes de vitesse et d'occupation de la

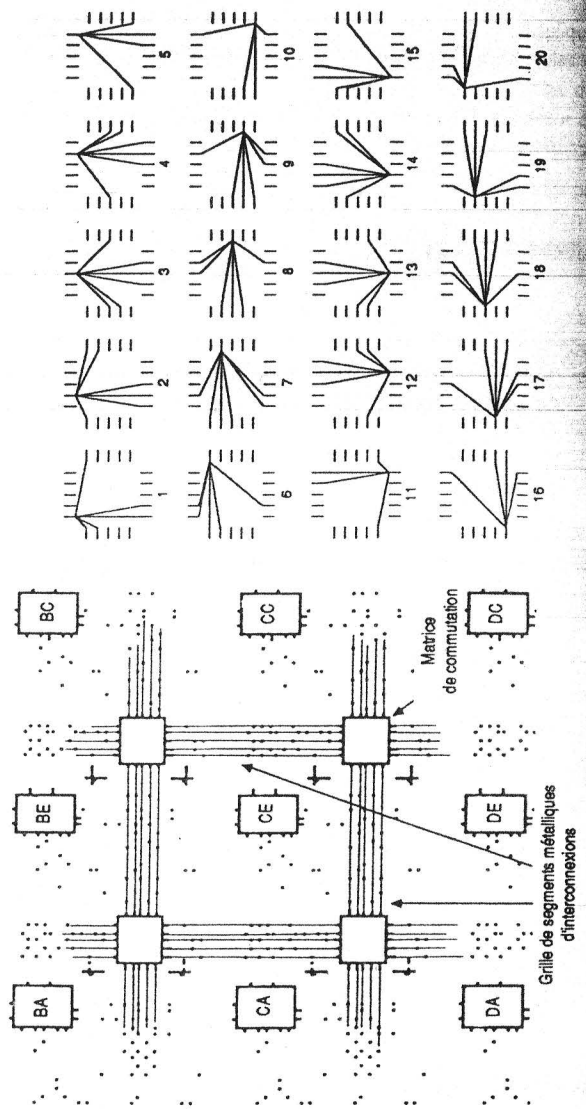


Figure 6.9 :
Emplacements et possibilités des matrices de commutation.

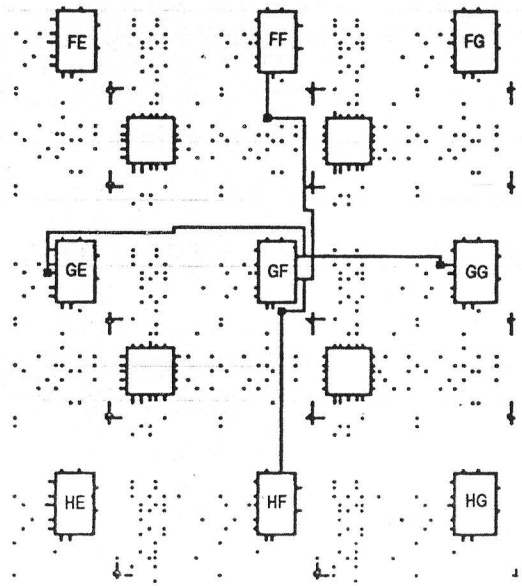


Figure 6.10 :
Utilisation des possibilités d'interconnexions directes.

puce. Elles résultent du fait que, en raison de la géométrie des cellules, il est possible de relier directement certaines entrées de l'une aux sorties de l'autre sans faire appel aux autres ressources d'interconnexions. Les liaisons qui peuvent être établies de cette façon sont les suivantes :

- Pour chaque CLB, la sortie X peut être reliée directement à l'entrée B du CLB situé immédiatement sur sa droite et à l'entrée C du CLB situé sur sa gauche. La sortie Y, quant à elle, peut être reliée directement de la même façon à l'entrée D du CLB du dessus ou à l'entrée A du CLB de dessous.
- Pour chaque CLB adjacent à un IOB, les connexions directes sont possibles tour à tour avec les entrées I ou les sorties O des IOB selon leur positionnement sur la puce.

Les lignes longues ou *long lines* sont des moyens d'interconnexions un peu différents de ce que nous venons de voir en ce sens qu'elles ne passent pas par les matrices de commutation. Ce sont des lignes

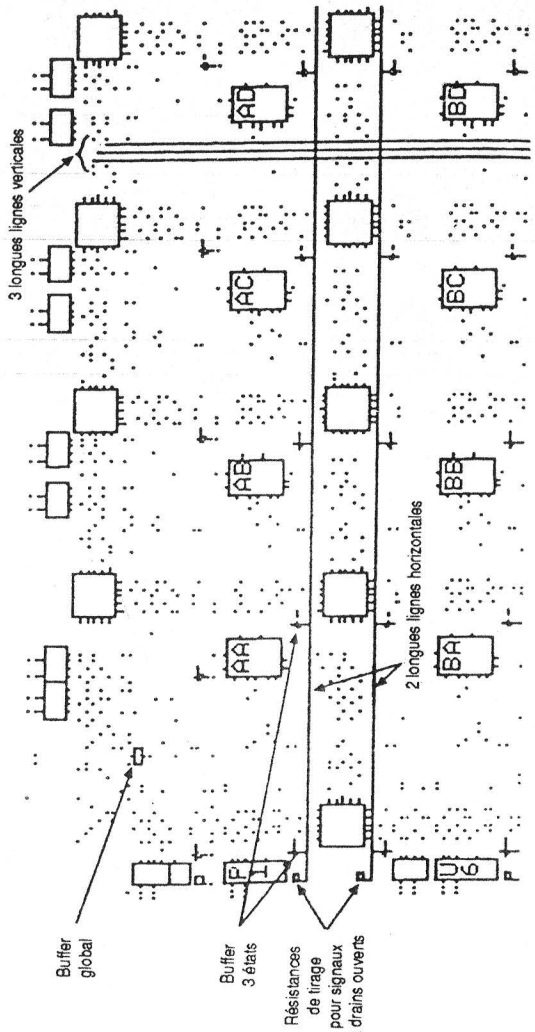


Figure 6.11 :
Emplacement des longues lignes.

métalliques qui traversent la puce de part en part, de haut en bas et de droite à gauche. Elles sont utilisées en priorité lorsque des signaux doivent être transmis avec un minimum de retard entre différents éléments afin d'assurer un synchronisme d'action aussi parfait que possible.

La figure 6.11 montre le positionnement de ces lignes longues qui sont au nombre de trois verticalement entre chaque colonne de CLB et de deux horizontalement entre chaque ligne de CLB. Deux lignes longues additionnelles sont placées à côté des matrices de commutation les plus externes et peuvent être scindées en deux parties pour former des demi-lignes longues.

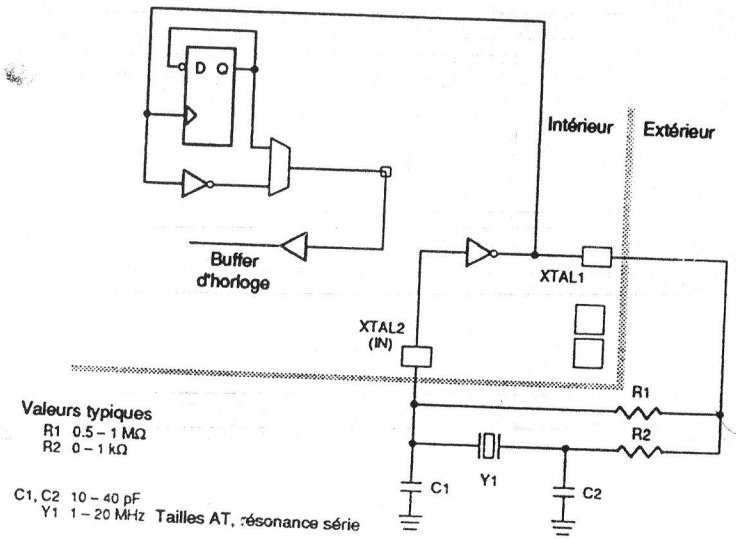
Ces lignes longues peuvent être pilotées par la sortie d'un IOB ou d'un CLB de façon traditionnelle, au moyen d'un buffer disponible, ou bien encore en mode bus par utilisation de buffers trois états également disponibles sur la puce.

Fort de toutes ces possibilités, le logiciel de placement routage, disponible avec les outils de développement, fait un choix qui se veut le plus judicieux possible. Comme ses critères de sélection ne sont pas nécessairement les vôtres, sachez qu'il est possible avec un éditeur graphique spécial, également fourni avec le logiciel de développement, d'agir vous-même sur la sélection des moyens d'interconnexions utilisés afin de privilégier un signal au timing critique par exemple.

L'OSCILLATEUR À QUARTZ

Comme de nombreuses applications logiques font appel à un oscillateur à quartz, Xilinx a prévu de doter ses LCA d'un amplificateur inverseur à haute vitesse, placé dans un angle de la puce, qui peut être activé lors de la phase de programmation pour réaliser un oscillateur à quartz. Il utilise alors les deux IOB voisins dans une configuration adaptée pour réaliser l'oscillateur dont le schéma vous est présenté figure 6.12.

Attention ! Deux points importants sont à noter à propos de cet oscillateur. Tout d'abord il ne peut être réalisé que dans un *angle* de la puce où se trouve l'amplificateur haute vitesse prévu à cet effet. Cet *angle* dépend du type de boîtier utilisé comme indiqué sur le tableau de cette même figure.



	44 PIN	68 PIN	84 PIN		100 PIN		132 PIN	160 PIN	164 PIN	175 PIN
	PLCC	PLCC	PLCC	PGA	CQFP	PQFP	PGA	PQFP	CQFP	PGA
XTAL 1 (OUT)	30	47	57	J11	67	82	P13	82	105	T14
XTAL 2 (IN)	26	43	53	L11	61	76	M13	76	99	P15

Figure 6.12 :
Réalisation d'un oscillateur à quartz avec un LCA.

Le deuxième point à prendre en compte est que si cet oscillateur n'est pas utilisé, les deux IOB qui l'entourent sont alors utilisables tout à fait normalement et au même titre que les autres ressources du boîtier.

LA PROGRAMMATION DES LCA

Comme nous l'avons expliqué en introduction, un LCA se *programme* en recopiant dans sa RAM interne la configuration des interconnexions à réaliser. Le format de ces informations est très particulier et ne vous concerne pas puisqu'il est produit automatiquement par le logiciel de développement sous forme d'un flot de données qu'il ne reste plus qu'à envoyer au LCA.

Ce que vous devez savoir, par contre, c'est comment fournir ce flot de données et à partir de quoi, et ce, d'autant que si vous n'utilisez pas de pile de sauvegarde dans le support comme nous l'avons expliqué en introduction, il faut reconfigurer votre LCA à chaque mise sous tension.

Pour ce faire, les LCA disposent de trois pattes, M_0, M_1 et M_2 qui servent à définir le mode de chargement suite à la mise sous tension conformément aux indications de la figure 6.13. Nous allons voir en détail à quoi correspondent ces divers modes et ce d'autant qu'ils conduisent à des schémas différents.

M0	M1	M2	Mode	Données
0	0	0	Maître	Séries
0	0	1	Maître	Parallèles - Départ de 0000
0	1	0	Ne pas utiliser	–
0	1	1	Maître	Parallèles - Départ de FFFF
1	0	0	Ne pas utiliser	–
1	0	1	Périphérique	Parallèles
1	1	0	Ne pas utiliser	–
1	1	1	Esclave	Séries

Figure 6.13 :
Tableau de sélection des modes de programmation d'un LCA.

Le premier mode est le plus simple et le moins encombrant en termes d'occupation de circuit imprimé. Il fait appel à une mémoire à accès série telle, par exemple celles commercialisées par Xilinx sous les références XC 1736 et XC 1765. Cette mémoire de type OTP a été préalablement programmée sur le système de développement utilisé pour le LCA et n'a donc plus qu'à être recopiée dans le LCA. Cela se produit en utilisant le schéma de la figure 6.14.

Comme on le voit sur cette figure, le LCA à configurer génère tous les signaux de dialogue nécessaires pour la PROM et, lorsqu'il a fini de recopier son contenu dans sa RAM interne, il le signale à la circuiterie dans laquelle il est utilisé au moyen de la ligne DONE. Comme

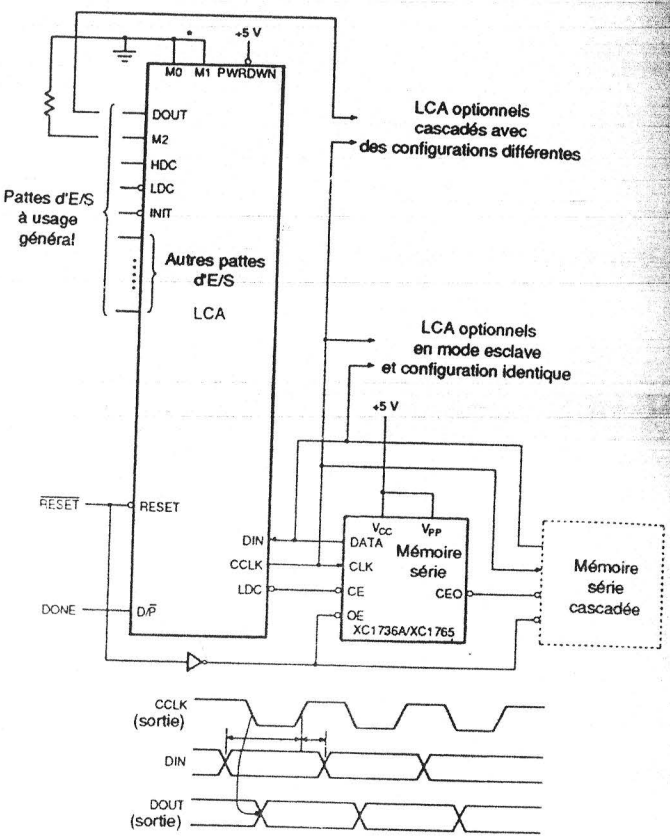


Figure 6.14 : Schéma de programmation en mode maître série.

le laisse supposer également cette même figure, une seule PROM peut être utilisée pour configurer plusieurs LCA qui sont alors cascadés. Le premier, qui est celui de la figure, est alors le maître et génère l'horloge pour les autres qui sont des esclaves. Les données issues de la PROM et destinées aux autres LCA leur sont envoyées par la sortie DOUT de ce premier LCA. La figure 6.17 qui présente le principe de connexion en mode esclave explique cela aussi bien qu'un long discours.

Les deux modes suivant sont analogues au précédent mais utilisent des données fournies sous forme parallèle, c'est-à-dire des données qui peuvent provenir d'une mémoire PROM classique (UVPROM ou EEPROM) par exemple.

Le principe de connexion du LCA et de la mémoire est alors celui indiqué figure 6.15. Il fait évidemment appel à plus de pattes puisque le LCA doit disposer de 8 lignes de données mais aussi de N lignes d'adresses nécessaires pour balayer la mémoire qui est de type classique rappelons-le.

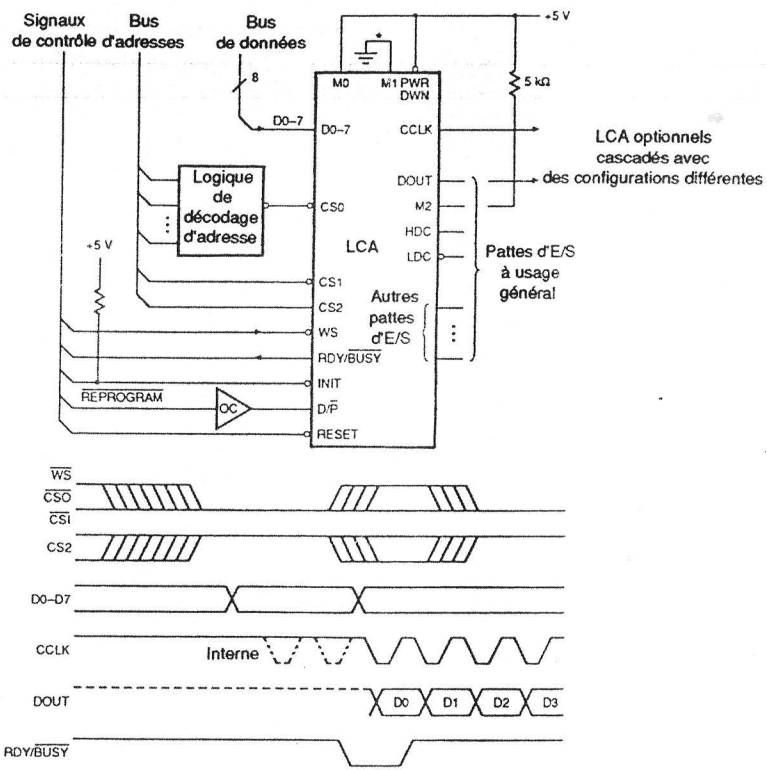


Figure 6.15 : Schéma de programmation en mode maître parallèle.

Comme dans le cas précédent, une mise en cascade des LCA est possible et la fin de la phase de configuration est signalée grâce à la ligne DONE. Précisons également que, lorsque cette phase de configuration est terminée, les lignes utilisées comme adresses et données pour la connexion à la mémoire reprennent leurs rôles normaux d'entrées/sorties logiques. Il faut toutefois faire attention lors de la conception du circuit, à ce que, pendant la phase de configuration, les

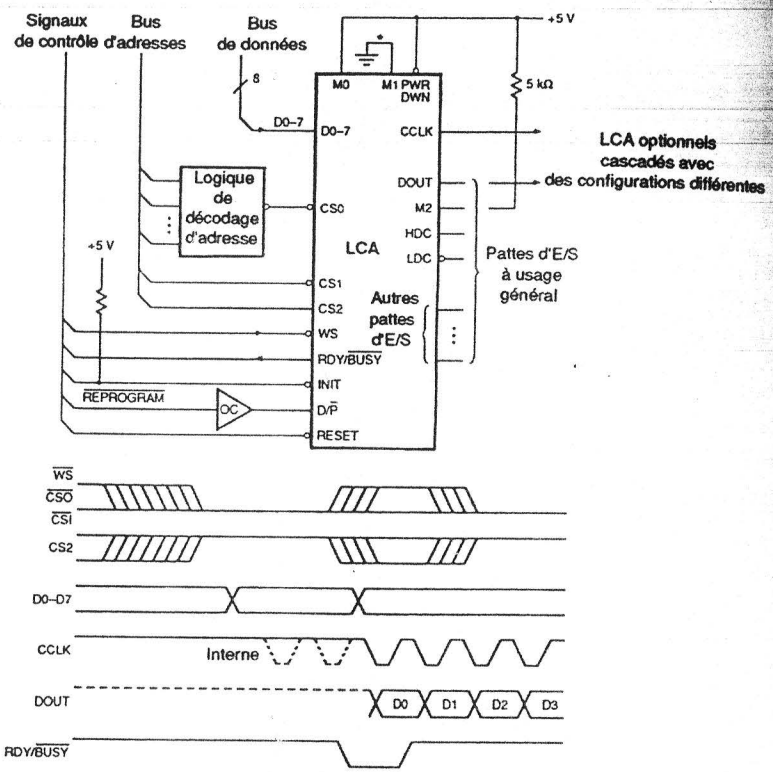


Figure 6.16 :
Schéma de programmation en mode périphérique.

circuits éventuellement connectés sur ces pattes ne puissent interférer avec les données lues dans la mémoire ou avec les adresses qui lui sont appliquées.

Ces deux modes permettent donc l'utilisation d'une mémoire ordinaire mais ils consomment plus de place sur le circuit imprimé (boîtier 28 pattes pour la mémoire normale contre 8 pattes pour la PROM série Xilinx) et ils compliquent parfois le dessin de la circuiterie avoisinante.

Remarquez que ces deux modes diffèrent pas le sens d'évolution des adresses qui, dans un cas partent de 0000 et croissent, alors que dans l'autre cas elles partent de FFFF et diminuent. Ces différences ont été prévues pour assurer une compatibilité avec les différents types de microprocesseurs dont certains commencent l'exécution des programmes par les adresses hautes et d'autres par les adresses basses.

Le mode suivant est le mode passif de type périphérique. Il a été prévu pour que le LCA puisse être configuré à partir d'un microprocesseur quelconque comme s'il en était un simple périphérique. La figure 6.16 vous montre le schéma utilisé dans ce cas.

Le bus de données est appliqué aux entrées de données du LCA ; les lignes d'adresses sont décodées et sont utilisées pour valider le boîtier dans l'espace mémoire du microprocesseur et quelques lignes annexes permettent le contrôle du processus de programmation. Les données chargées dans la RAM du LCA sont alors lues quelque part dans la ROM de programme du microprocesseur pour être téléchargées dans le LCA. Cette configuration est particulièrement utile pour les applications dans lesquelles le contenu du LCA est susceptible de changer en cours d'utilisation.

Le dernier mode est le mode esclave dans lequel le LCA reçoit ses données depuis un autre LCA de la façon schématisée figure 6.17 à titre d'exemple. Le LCA maître, quant à lui, peut être interfacé à une mémoire en mode parallèle ou série, ou même être utilisé en mode périphérique sans que cela modifie quoi que ce soit au niveau du câblage des esclaves. Ce mode esclave est particulièrement intéressant lorsque l'on est en présence d'un montage contenant plusieurs LCA car, au niveau des LCA esclaves, il conduit à un câblage et à des contraintes minimes.

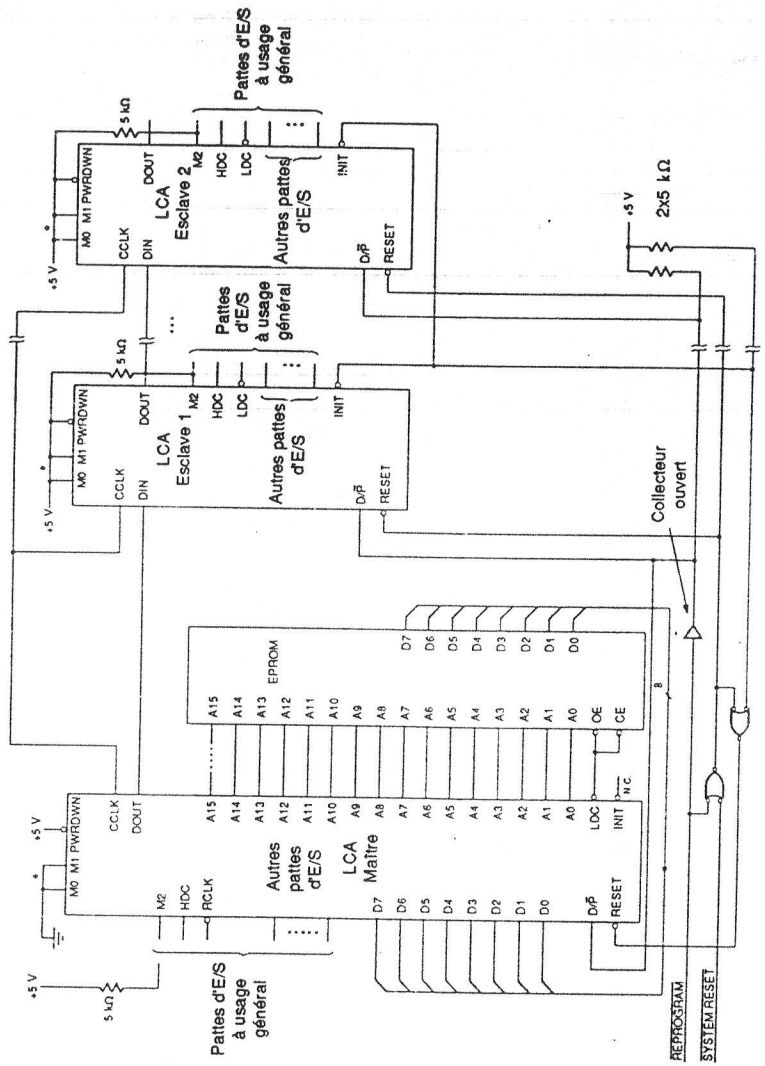


Figure 6.17 :
Schéma de programmation en mode esclave.

LES OUTILS DE DÉVELOPPEMENT

Comme l'a fait Altera pour les EPLD, Xilinx a mis sur le marché en même temps que ses premiers LCA un système de développement performant capable de fonctionner sur divers micro-ordinateurs dont les stations de travail Sun, Apollo et DEC bien sûr, mais aussi sur n'importe quel compatible PC à base de 386 ou au delà disposant de suffisamment de mémoire.

Ce logiciel, un peu limité à ses débuts, est maintenant entré dans l'âge adulte et permet de concevoir des LCA sans connaissance préalable du système et après une phase d'apprentissage de courte durée (une journée à deux jours environ).

Il est composé de trois sous-ensembles fonctionnels principaux :

- Le sous-ensemble d'entrée du *schéma*.
- Le sous-ensemble de compilation ou de placement routage dans le LCA.
- Le sous-ensemble de vérification et de programmation du circuit.

Le sous-ensemble d'entrée du schéma est en fait scindé en plusieurs logiciels divers que vous êtes libre d'acquérir ou non et qui réalisent en fait l'interfaçage entre tous les grands logiciels de saisie de schéma (Futurenet, Orcad, Mentor Graphics, etc.) et le logiciel de saisie de schéma Xilinx. Précisons que Xilinx fournit aussi une bibliothèque de fonctions ainsi que divers logiciels de traductions de fichiers au format PALASM en format Xilinx par exemple.

Vient ensuite le système de conception du LCA proprement dit qui, à partir des données d'entrée en provenance des divers logiciels précédents, élabore la configuration des divers IOB et CLB ainsi que les interconnexions nécessaires. Ce logiciel peut travailler seul grâce à une fonction APR qui signifie *Automatic Place and Route*, mais il peut aussi travailler de manière interactive ce qui vous permet par exemple de réaliser vous-même certaines connexions critiques.

À la fin du traitement réalisé par ce logiciel, le fichier de configuration du LCA est disponible et, si vous ne souhaitez pas passer par une phase de simulation, vous pouvez directement programmer votre circuit. En effet, un câble qui se connecte sur le port série du micro-ordinateur est fourni et, moyennant une interface de raccordement fort

simple au LCA à tester ; il vous permet d'essayer celui-ci immédiatement sans passer par la phase recopie de PROM.

Comme pour les EPLD et parce que c'est encore plus nécessaire pour les LCA dont la configuration interne peut être très complexe, diverses solutions de simulation sont disponibles, toujours sous forme de logiciels. Ils offrent une simulation fonctionnelle et temporelle du circuit avec la possibilité de visualiser l'évolution de points internes du circuit ce qui permet en général de venir rapidement à bout des problèmes les plus délicats.

Ici encore, et comme pour le logiciel commercialisé par Altera pour les EPLD, la prise en main du produit, au moins pour ce qui est des phases d'entrée de schéma et de configuration du LCA, est particulièrement simple et rapide et ne demande qu'un minimum d'apprentissage. L'utilisation de l'écran en mode graphique, de la souris et des menus déroulants facilitent bien évidemment grandement les choses. Le logiciel de simulation quant à lui est un peu plus délicat à maîtriser, non pas parce qu'il est mal conçu, mais en raison de ses nombreuses possibilités.

Aujourd'hui, Xilinx n'est évidemment plus le seul fournisseur d'outils de développement pour ses LCA, et tous les grands fabricants de programmateurs ou de logiciels de dessin de schémas et de circuits imprimés proposent des produits capables de travailler avec les LCA.

LES DIFFÉRENTES FAMILLES DE LCA

Les LCA sont actuellement divisés en trois familles :

- La famille 2000 qui est la plus ancienne et qui comporte quatre références.
- La famille 3000 qui est la plus présente sur le marché à l'heure actuelle et qui comporte vingt et une références.
- La famille 4000 qui est la plus récente et qui comporte, au moment où ces lignes sont écrites, une dizaine de références.

Nous n'allons pas détailler ici chaque circuit de ces familles puisque tous respectent les principes généraux que nous venons d'exposer largement ci-avant. Nous nous bornerons donc à présenter les tailles des divers circuits ainsi que les schémas des CLB et IOB des familles 2000 et 4000.

En effet, dans l'exposé précédent nous avons utilisé à titre d'exemple les schémas des CLB et IOB de la famille 3000 puisque c'est la plus répandue à l'heure actuelle. En fait, les CLB et IOB de la famille 2000, qui est la plus ancienne, sont un peu plus simples tandis que les mêmes éléments mais de la famille 4000 sont un peu plus complexes puisque cette famille est la plus récente. Les principes et fonctions restent cependant identiques et vous n'aurez donc aucune difficulté à analyser ces schémas.

La famille 2000

Cette famille, qui fait un peu figure de parent pauvre en 1995, ne doit pas être traitée avec dédain ; il faut en effet se souvenir qu'elle a été introduite sur le marché en 1984 et que, à l'époque, elle a constitué une véritable petite révolution.

Elle comporte seulement deux membres principaux (les variantes affublées du suffixe L étant des versions 3,3 volts des mêmes circuits) que sont le XC 2064 et le XC 2018 dont les principales caractéristiques sont résumées figure 6.18. L'IOB, plus simple que celui de la famille 3000, vous est présenté figure 6.19.

Référence	V_{CC}	Capacité Logique (nombre de portes)	CLB	E/S utilisateur	Bits de configuration
XC 2064	5,0 V	800-1 000	64	58	12 038
XC 2064 L	3,3 V	800-1 000	64	58	12 038
XC 2018	5,0 V	1 200-1 500	100	74	17 878
XC 2018	3,3 V	1 200-1 500	100	74	17 878

Figure 6.18 : Les LCA de la famille 2000.

Vous y reconnaissez aisément les prémices de la famille 3000 comme d'ailleurs dans le CLB présenté figure 6.20.

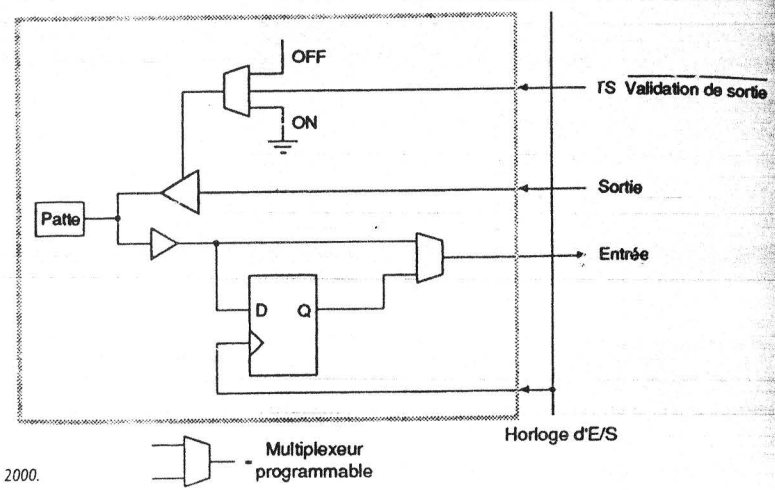


Figure 6.19 :
Schéma des IOB de la famille 2000.

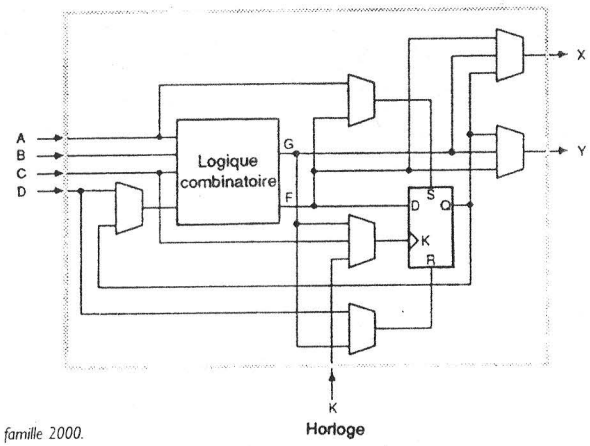
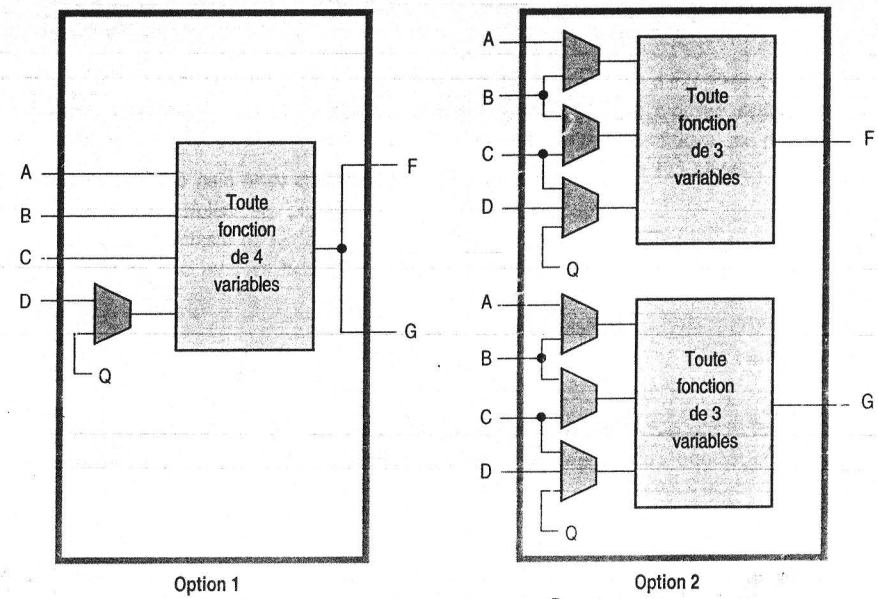


Figure 6.20 :
Schéma des CLB de la famille 2000.

Les options de configuration de ce CLB sont également un peu moins nombreuses que pour celui de la famille 3000 comme le montre la figure 6.21. Ceci n'est cependant pas très contraignant et de nombreuses cartes d'interface complexes pour compatibles PC fonctionnent actuellement avec de *simples* 2018 ou 2064.



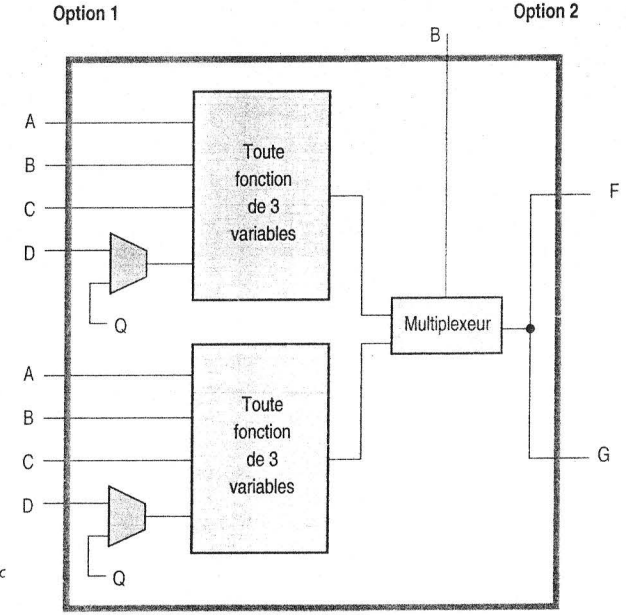


Figure 6.21 :
Les différents modes de fonctionnement du bloc logique d'entrée du CLB de la famille 2000.

Ces circuits existent dans divers types de boîtiers LCC ou PGA et avec des fréquences maximum de travail, mesurées au niveau des bascules internes, qui vont de 50 à 100 MHz selon les versions.

La consommation de tels circuits varie bien évidemment avec la fréquence de travail et le nombre de cellules et d'interconnexions internes utilisées. Au repos, elle est au maximum de 12 mA et peut atteindre les 100 mA à 50 MHz et avec un taux de remplissage normal du boîtier.

La famille 3000

Cette famille qui est la plus utilisée actuellement comporte vingt et une références qui vous sont présentées figure 6.22.

Référence	CLB	Réseau	E/S utilisateur	Bascules	Longues lignes horizontales	Bits de configuration
XC 3020, 3020 A, 3020 L, 3120	64	8 X 8	64	256	16	14 779
XC 3030, 3030 A, 3030 L, 3130	100	10 X 10	80	360	20	22 176
XC 3042, 3042 A, 3042 L, 3142	144	12 X 12	96	480	24	30 784
XC 3064, 3064 A, 3064 L, 3164	224	16 X 14	120	688	32	46 064
XC 3090, 3090 A, 3090 L, 3190	320	16 X 20	144	928	40	64 160
XC 3195	484	22 X 22	176	1 320	44	94 984

Figure 6.22 :
Les LCA de la famille 3000.

Comme vous pouvez le constater, la densité d'intégration est supérieure à celle des boîtiers précédents pour atteindre les 9000 portes pour la référence la plus dense. Nous vous laissons le soin de méditer ces chiffres quitte à regarder à nouveau les schémas des CLB présentés figure 6.6 qui sont au nombre de 484 dans un tel boîtier et les IOB présentés figure 6.5 qui sont au nombre de 176 !

Ces circuits sont disponibles eux aussi dans divers types de boîtiers LCC, PGA ou flat pack allant de 68 à 100 pattes selon les versions de boîtiers et les types de LCA.

Malgré l'accroissement de la densité d'intégration, la vitesse des circuits de cette famille ne s'est pas trouvée réduite, au contraire, puisque mesurée dans les mêmes conditions que pour les circuits de la série 2000, elle varie de 70 MHz à 125 MHz.

La famille 4000

Certains concurrents de Xilinx et non des moindres, avaient prévu un tassement, voire un blocage vers le haut, de la croissance des LCA du à des problèmes d'intégration. La famille 4000, récemment introduite sur le marché, leur donne tort avec brio.

En effet, comme le montre la figure 6.23, cette famille ne comporte pas moins de dix références distinctes dont la plus dense contient environ 20000 portes c'est à dire 900 CLB et 240 IOB plus quelques fonctions annexes qui n'existaient pas sur les versions précédentes.

Référence	XC4002	4003	4004	4005	4006	4008	4010	4013	4016	4020
Nombre de portes	2,000	3,000	4,000	5,000	6,000	8,000	10,000	13,000	16,000	20,000
Matrice de CLB	8 x 8	10 x 10	12 x 12	14 x 14	16 x 16	18 x 18	20 x 20	24 x 24	26 x 26	30 x 30
Nombre de CLB	64	100	144	196	256	324	400	576	676	900
Taille RAM	2,048	3,200	4,608	6,272	8,192	10,368	12,800	18,432	21,632	28,800
Nombre d'IOB	64	80	96	112	128	144	160	192	208	240

Figure 6.23 :
Les LCA de la famille 4000.

Comme si cela ne suffisait pas aux concepteurs de cette famille, ils ont un peu modifié le schéma des CLB qui prend maintenant l'allure visible figure 6.24. On y retrouve les mêmes principes de base que dans les versions précédentes mais avec un nombre d'entrées augmenté et des possibilités de configuration plus nombreuses.

Les IOB ont également subi un petit *lifting* pour devenir ce qui vous est présenté figure 6.25.

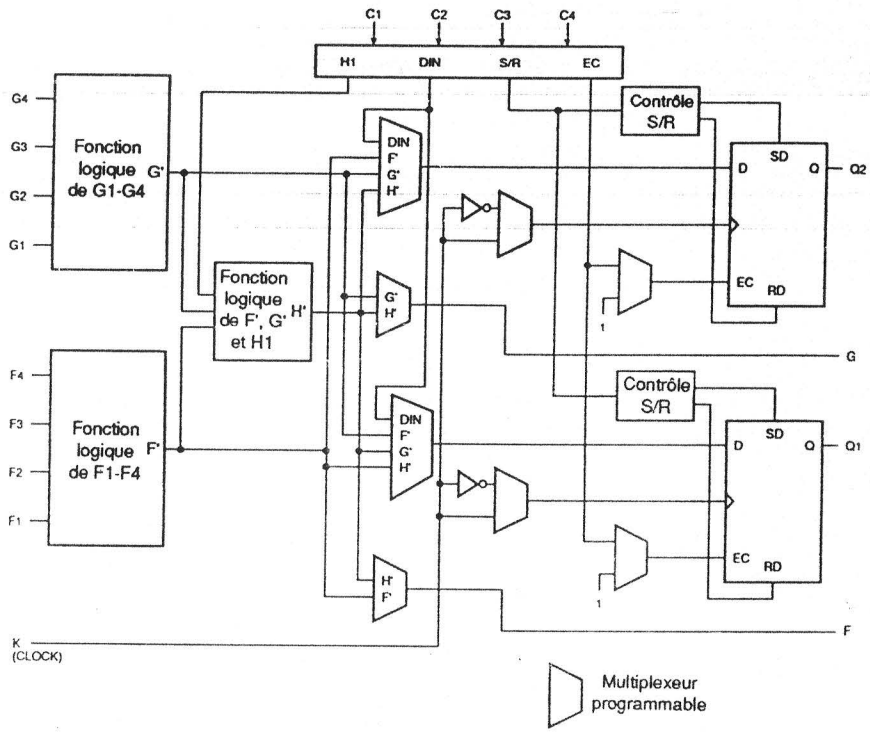


Figure 6.24 :
Schéma d'un CLB de la famille 4000.

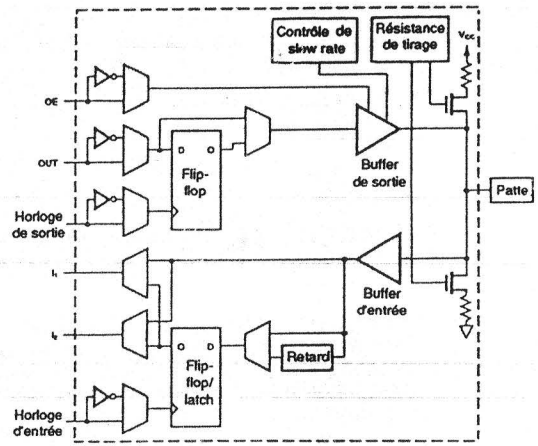


Figure 6.25 :
Schéma d'un IOB de la famille 4000.

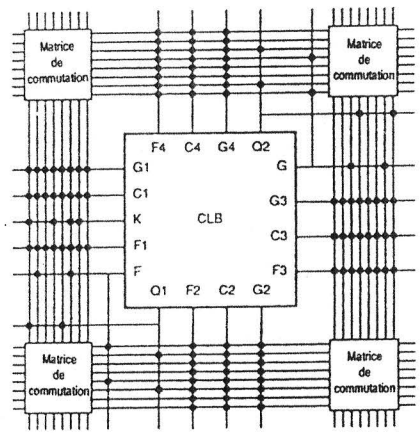


Figure 6.26 :
Accroissement des possibilités de connexion des CLB dans la famille 4000.

Les possibilités de connexions, tout en respectant ce que nous avons vu ci-avant avec les autres familles de LCA, ont également été améliorées. Comme le montre la figure 6.26 les possibilités de connexions directes au niveau des CLB ont été étoffées. De même, les différentes configurations possibles au niveau des matrices d'interconnexions entre les segments verticaux et horizontaux ont été uniformisées comme le montre la figure 6.27.

Toutes ces modifications sont le fruit de l'expérience acquise par Xilinx en plusieurs années de commercialisation de LCA et également suite aux critiques ou remarques formulées par les utilisateurs des produits.

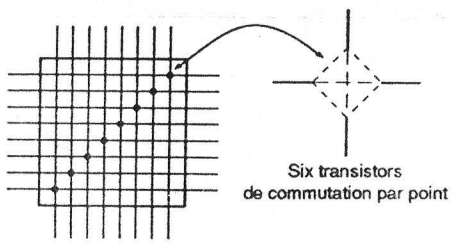


Figure 6.27 :
Uniformisation des possibilités de la matrice de commutation dans la série 4000.

LES FLEX D'ALTERA

Après avoir commercialisé pendant plusieurs années les EPLD de divers types vus au chapitre précédent, Altera a introduit sur le marché sa famille FLEX qui utilise, comme les LCA de Xilinx, une RAM pour stocker la configuration des connexions établies.

Les mêmes causes produisant les mêmes effets, les circuits FLEX d'Altera sont donc programmables en circuit en une centaine de ms, mais perdent la mémoire de leur configuration dès qu'ils ne sont plus alimentés. Cette configuration peut se faire sous forme série par recopie du contenu d'une EPROM, qui a pour nom EPC 1213 ou EPC 1064 chez Altera, ou sous forme parallèle par recopie d'une PROM plus classique ou à partir de la mémoire d'un système à base de microprocesseur.

La similitude avec les produits Altera s'arrête là car les circuits FLEX adoptent une architecture interne assez différente; très proche comme vous allez vous en rendre compte tout de suite de ce que nous avons vu pour la famille des EPLD MAX 9000 du même fabricant ce qui est, somme toute, assez logique.

Pour ce qui est de la densité d'intégration, les circuits FLEX arrivent à des tailles comparables à ceux de la plus grosse famille Xilinx qui est la famille 4000. En effet, comme le montre la figure 6.28, l'EPF 81500 intègre 32000 portes dont 16000 sont à la disposition de l'utilisateur.

	EPF8282 EPF8282V EPF8282A EPF8282AV	EPF8452 EPF8452A	EPF8636A	EPF8820 EPF8820A	EPF81188 EPF81188A	EPF81500 EPF81500A
Nombre de portes	5,000	8,000	12,000	16,000	24,000	32,000
Nb de portes utilisables	2,500	4,000	6,000	8,000	12,000	16,000
Nombre de bascules	282	452	636	820	1,188	1,500
Nombre de LE	208	336	504	672	1,008	1,296
Nombre maxi d'E/S	78	120	136	152	184	208
Boîtiers	84-pin PLCC 100-pin TQFP	84-pin PLCC 160-pin PQFP 160-pin PGA 100-pin TQFP	84-pin PLCC 160-pin PQFP 192-pin PGA 208-pin RQFP	160-pin PQFP 192-pin PGA 208-pin RQFP 225-pin BGA	208-pin PQFP 232-pin PGA 240-pin RQFP	240-pin RQFP 280-pin PGA 304-pin RQFP

Figure 6.28 :
Les différents circuits de la famille FLEX 8000 d'Altera.

ARCHITECTURE INTERNE

La figure 6.29 présente l'architecture interne des circuits FLEX 8000. L'air de famille avec la figure 5.26 des MAX 9000 est évident!

L'élément de base des circuits FLEX est le LE pour *Logic Element* dont nous verrons les possibilités dans un instant. Ces LE sont groupés par blocs de 8 afin de former un LAB ou... *Logic Array Bloc* bien sûr ! Ces LAB sont placés au sein d'un quadrillage réalisé par les lignes d'interconnexions rapides verticales et horizontales qui aboutissent chacune sur les IOE pour *Input Output Element*. Jusque là notre FLEX a un air de MAX 9000 au remplacement près des macro-cellules par les fameux LE.

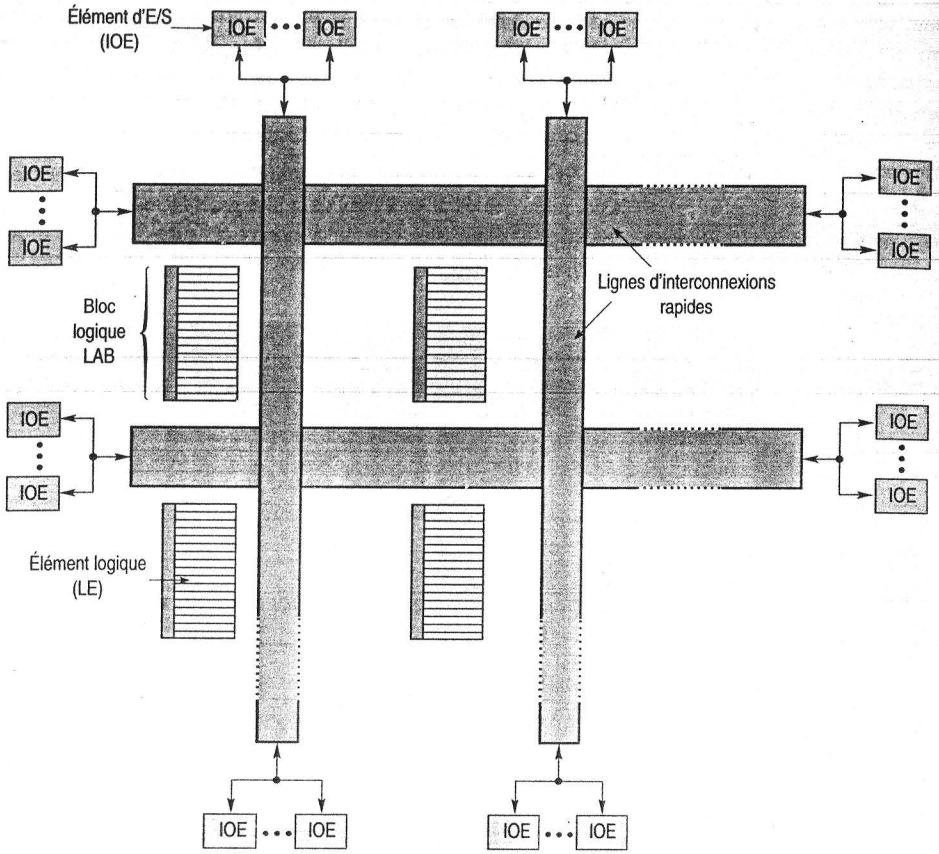


Figure 6.29 :
Architecture interne des circuits FLEX.

Comme le montre la figure 6.30, ce LE contient un bloc à 4 entrées appelé LUT pour *Look Up Table*, capable de générer n'importe quelle fonction combinatoire de 4 variables. Il est suivi d'une bascule pouvant être configurée en mode D, T, JK ou SR ou pouvant être éliminée. L'horloge et les mises à zéro et à un de cette bascule disposent de leurs propres lignes pouvant être pilotées par des entrées dédiées, des entrées/sorties à usage général ou de la logique interne.

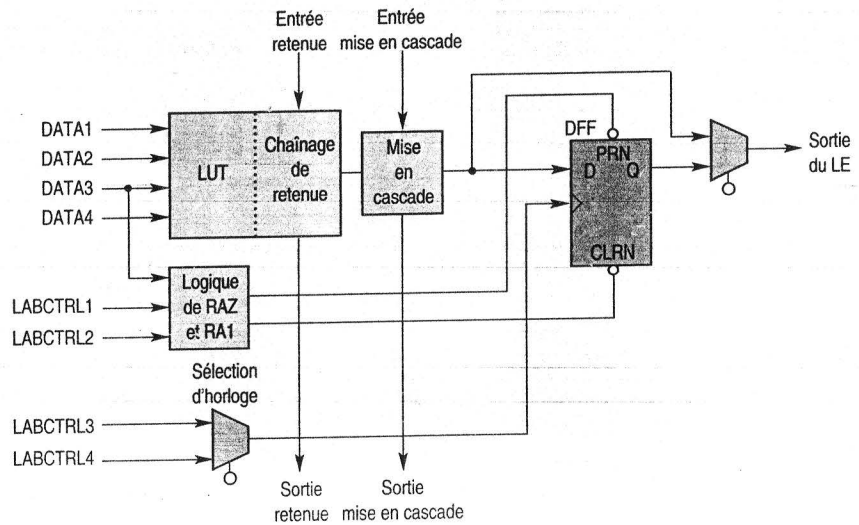


Figure 6.30 :
Structure interne d'un LE.

La présence de deux autres éléments est à noter : la *carry chain* ou chaînage de retenue et la *cascade chain* ou mise en cascade des LE. Ces deux liaisons ne passent pas par les systèmes d'interconnexions habituels et sont prévues pour cascader les LE lorsque des chronogrammes critiques doivent être respectés : compteurs rapides et additionneur pour le chaînage des retenues, fonctions de nombreuses entrées parallèles simultanées pour la mise en cascade. Leur utilisation doit évidemment n'avoir lieu que lorsque c'est strictement nécessaire car cela limite la souplesse de routage des liaisons sur la puce.

Ceci étant, ce *Logic Element* peut être configuré de quatre façons différentes présentées figure 6.31. Chacune correspond à un type d'application logique bien précis comme indiqué sur cette même figure.

La circuiterie de mise à zéro ou à un de la bascule, quant à elle, respecte le schéma présenté figure 6.32. Ici encore, on bénéficie d'une grande souplesse de câblage.

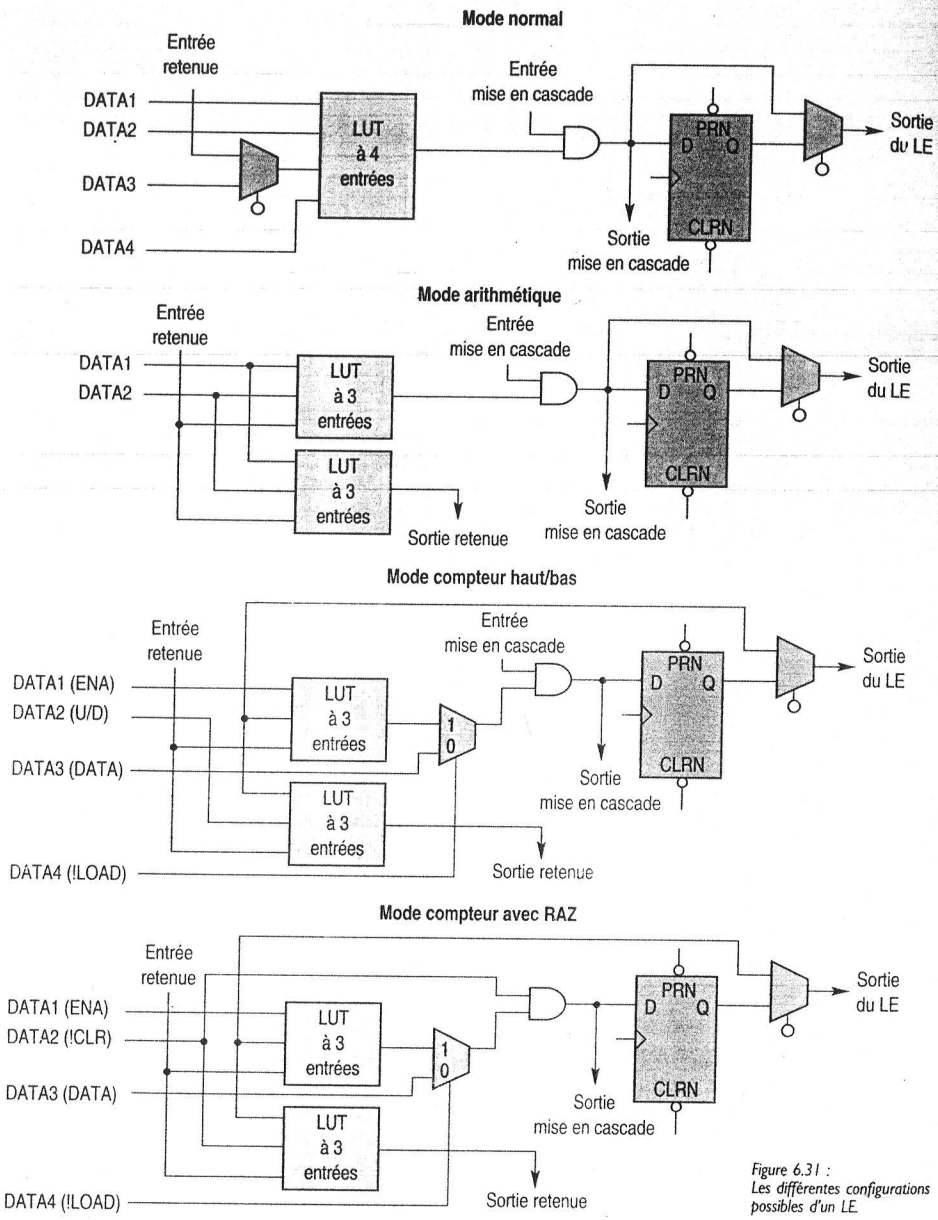


Figure 6.31 :
Les différentes configurations possibles d'un LE.

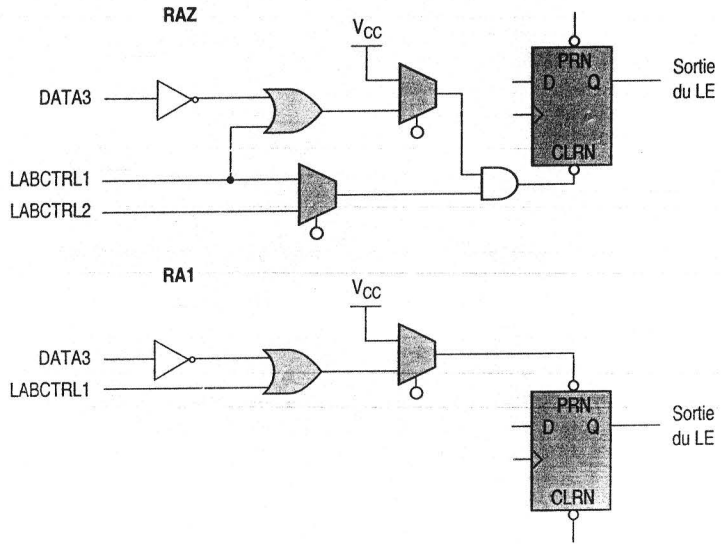


Figure 6.32 :
Circuiterie de mise à zéro ou à un de la bascule d'un LE.

Un LAB regroupe, comme nous l'avons dit, 8 LE selon le principe présenté figure 6.33. Chaque bloc d'interconnexion de LAB fournit 4 signaux aux LE. Ces signaux peuvent provenir d'entrées dédiées, d'entrées/sorties à usage général ou d'autres éléments logiques du circuit via justement ce bloc d'interconnexion.

Les sorties des LE aboutissent à la circuiterie de connexion aux lignes d'interconnexions rapides verticales et horizontales selon le schéma visible figure 6.34. Chaque sortie de LE pilote 2 lignes verticales et peut également piloter une ligne horizontale via un multiplexeur programmable. Ces sorties reviennent aussi bien évidemment sur la zone d'interconnexion propre au LAB où se trouve le LE concerné. Remarquez que le multiplexeur programmable pouvant relier la sortie du LE aux lignes horizontale permet aussi de relier 2 lignes verticales à 2 lignes horizontales.

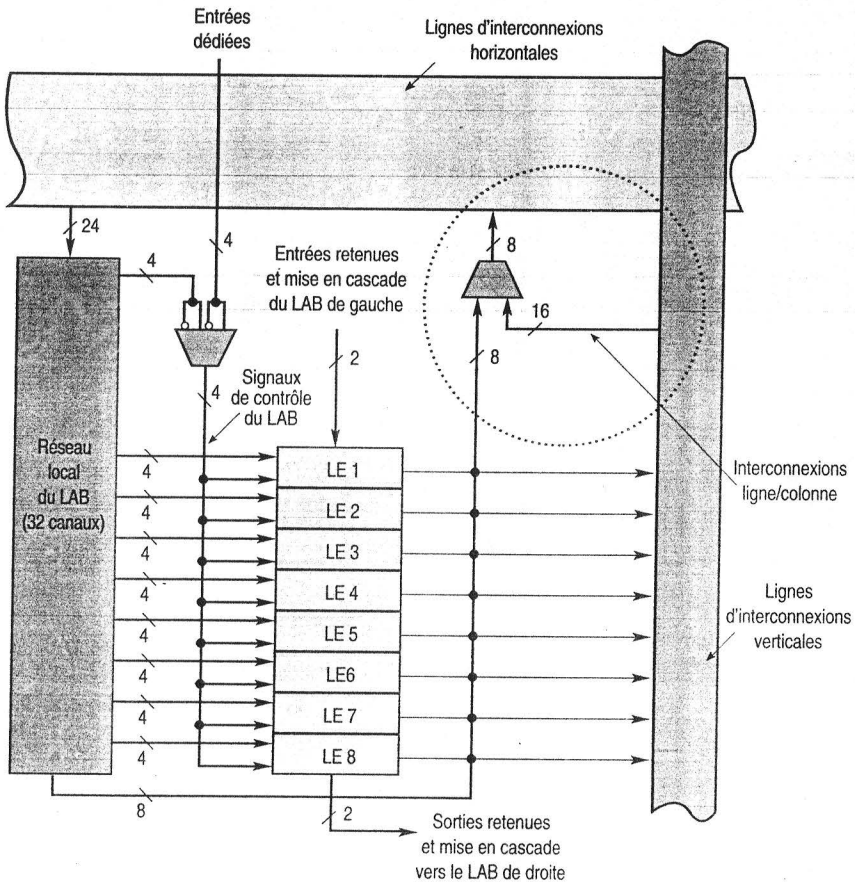


Figure 6.33 :
Principe de regroupement des LE au sein d'un LAB.

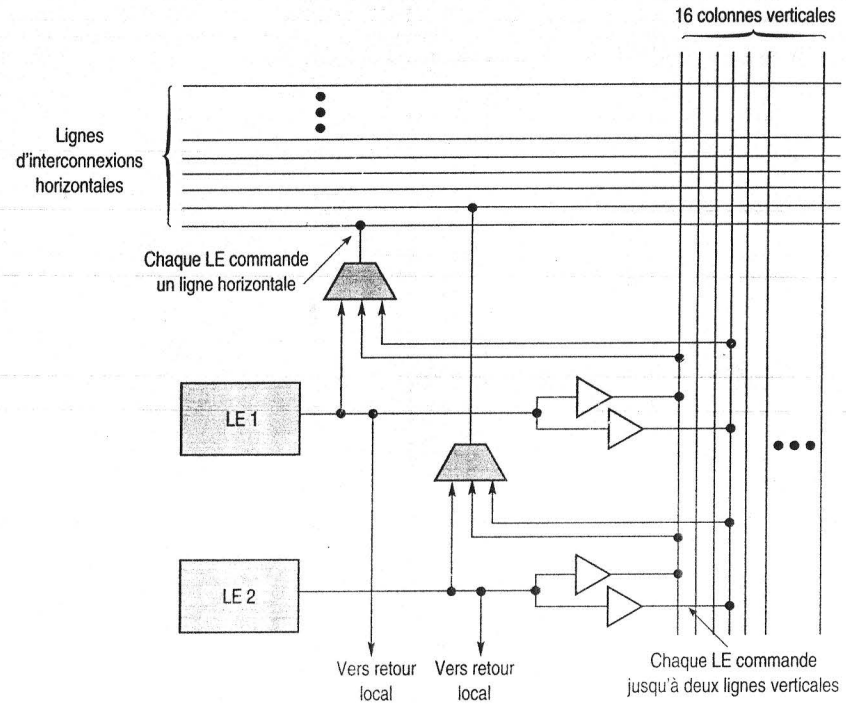


Figure 6.34 :
Détail des connexions possibles entre les LE et les lignes d'interconnexions rapides.

Comme nous l'avons vu en figure 6.29, ces lignes d'interconnexions rapides aboutissent sur les IOE ou *Input Output Elements*. Ceci a lieu selon le schéma présenté figure 6.35 pour ce qui est des lignes horizontales étant entendu que chaque groupe de lignes arrive sur 8 IOE.

Tout les IOE placés en entrées sont capables de piloter simultanément 2 lignes d'interconnexions rapides horizontales. Les IOE placés en sortie par contre sont pilotés par une ligne d'interconnexion rapide horizontale choisie parmi celles disponibles grâce à un multiplexeur programmable. Il y a évidemment autant de multiplexeurs que d'IOE et chacun à un nombre d'entrées égal au nombre de lignes d'interconnexions rapides horizontales divisé par ce nombre d'IOE. Autant dire que l'on ne peut pas, en pratique, relier n'importe quelle ligne à

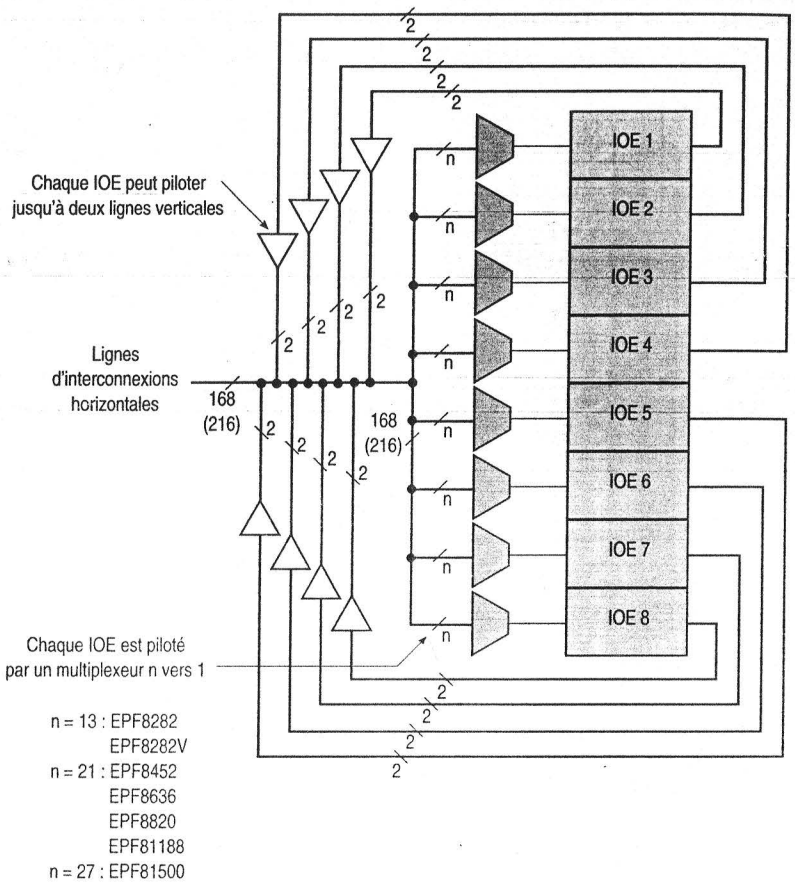


Figure 6.35 :
Principe de connexion entre les lignes d'interconnexions rapides horizontales et les IOE.

n'importe quel IOE. Si l'on impose un brochage au circuit, le logiciel de placement routage va donc faire le nécessaire pour que les *bons* signaux arrivent sur les lignes horizontales correspondant aux multiplexeurs reliés aux pattes du boîtier qui vous intéressent. Ce forçage du brochage peut donc compliquer le routage, voire dans des circuits très remplis le rendre impossible. C'est là une limitation commune à tous les FPGA.

Pour ce qui est des IOE relatifs aux lignes verticales, le principe est quasiment le même comme le montre la figure 6.36. Chaque groupe de 16 lignes arrive sur 2 IOE via 2 multiplexeurs pour ce qui est de la sortie. En entrée par contre, chaque IOE peut piloter 2 lignes verticales.

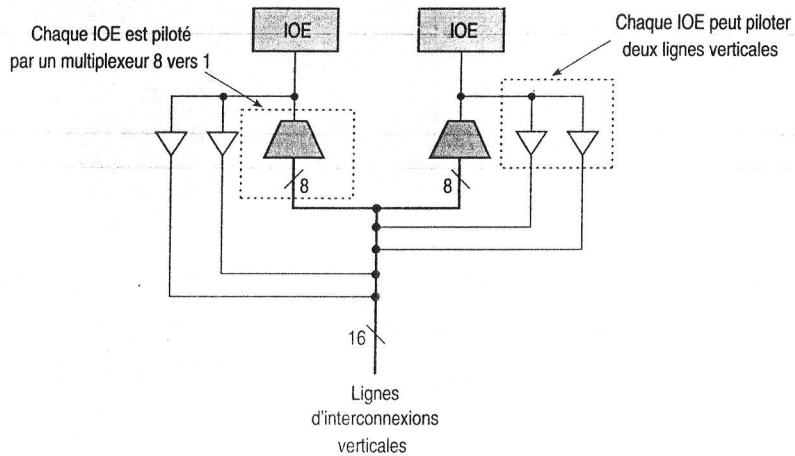


Figure 6.36 :
Principe de connexion entre les lignes d'interconnexions rapides verticales et les IOE.

Comme le montre la figure 6.37, ces IOE permettent toutes les combinaisons d'entrées ou de sorties souhaitables : directes ou à registre avec en plus une possibilité de contrôle du passage en troisième état. En outre on retrouve ici la possibilité de programmer le temps de montée que nous avions vue avec les EPLD MAX 7000 et MAX 9000.

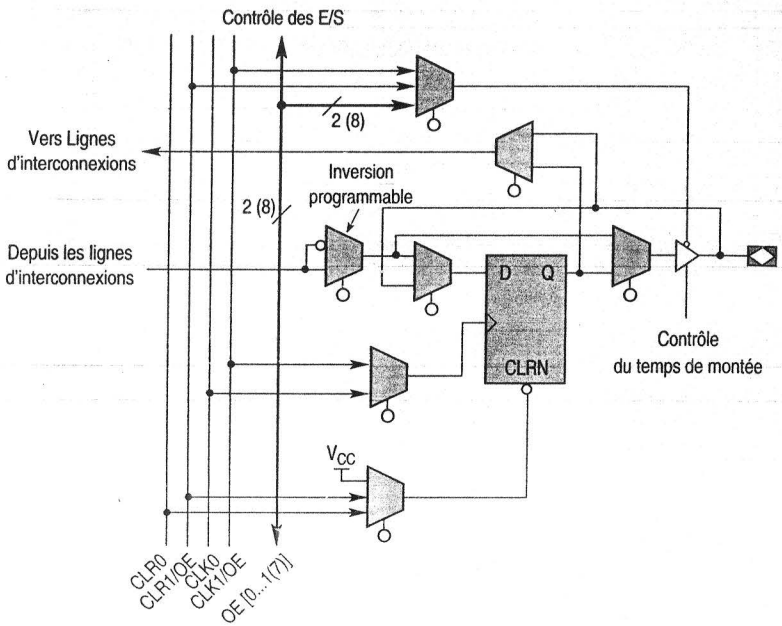


Figure 6.37 :
Schéma interne d'un IOE.

LA PROGRAMMATION DES FLEX

Comme pour les LCA de Xilinx, divers modes de programmation existent et sont sélectionnés par mise au niveau adéquat de trois pattes du boîtier.

Le mode le plus simple et le plus naturel est le mode actif série où, comme le montre la figure 6.38, une mémoire EPROM à accès série est recopiée dans le circuit FLEX à configurer. Ce mode est dit actif puisque c'est le circuit FLEX lui-même qui génère et fait progresser l'adressage de la mémoire.

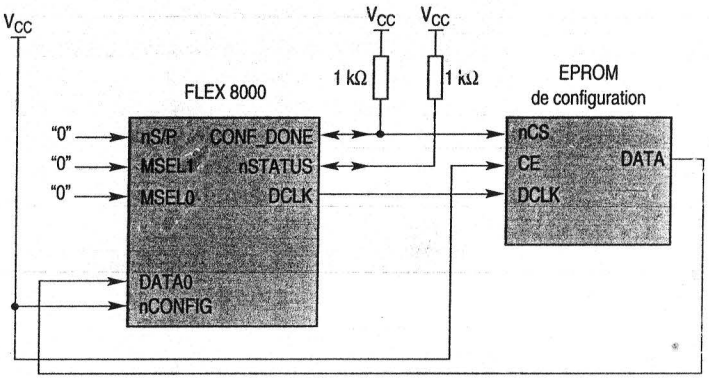


Figure 6.38 :
Programmation d'un FLEX en mode actif série.

Le mode actif parallèle, quant à lui, est présenté figure 6.39. Son principe est le même à savoir que le circuit FLEX génère l'adressage d'une mémoire externe, mais celle-ci est cette fois de type parallèle (normale si vous préférez). Comme pour les LCA pour lesquels ce mode existe aussi, il présente l'inconvénient par rapport au mode série de consommer de nombreuses pattes du FLEX, au moins pendant la phase de configuration.

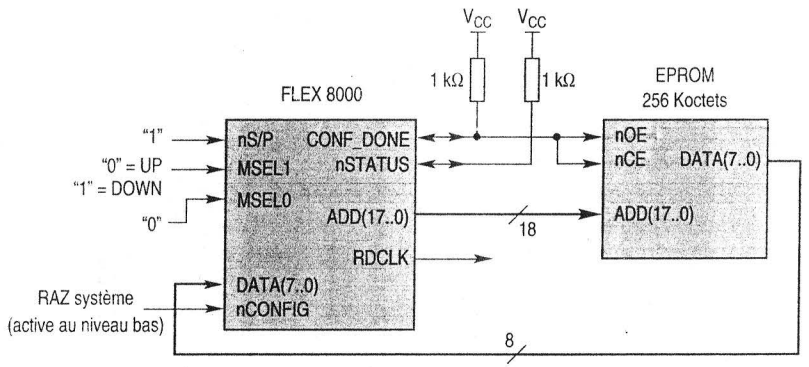


Figure 6.39 :
Programmation d'un FLEX en mode actif parallèle.

Précisons également que, lorsque cette phase de configuration est terminée, les lignes utilisées comme adresses et données pour la connexion à la mémoire reprennent leurs rôles normaux d'entrées/sorties logiques. Il faut toutefois faire attention lors de la conception du circuit à ce que, pendant la phase de configuration, les circuits éventuellement connectés sur ces pattes ne puissent interférer avec les données lues dans la mémoire ou avec les adresses qui lui sont appliquées.

Ce mode permet donc l'utilisation d'une mémoire ordinaire, mais il consomme plus de place sur le circuit imprimé (boîtier 28 ou 32 pattes pour la mémoire normale contre 8 pattes pour la PROM série) et il complique parfois le dessin de la circuiterie avoisinante.

Le mode passif parallèle synchrone, quant à lui, vous est présenté figure 6.40. Il est utilisable lorsqu'un microprocesseur ou microcontrôleur est présent dans le système. Ce dernier, via 4 lignes d'un port parallèle commande le circuit FLEX qui reçoit, par contre, ses données directement du bus du système. Il est évident qu'une telle approche n'est utilisable que si l'on a une maîtrise parfaite des chronogrammes du bus de données et de son parfait synchronisme avec l'horloge générée sur le port parallèle et appliquée à la ligne DCLK du FLEX.

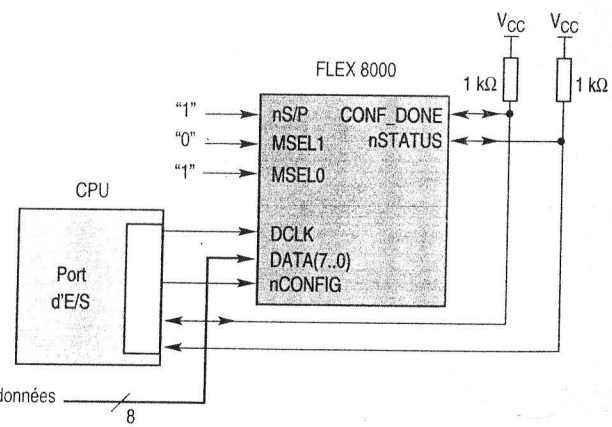


Figure 6.40 : Programmation d'un FLEX en mode passif parallèle synchrone.

Si tel n'est pas le cas, il est préférable de faire appel au mode passif parallèle asynchrone schématisé figure 6.41. Dans ce mode les lignes de contrôle et les données envoyées au FLEX sont fournies par un port parallèle piloté par le microprocesseur. Le synchronisme nécessaire entre données et horloge est alors très facile à respecter.

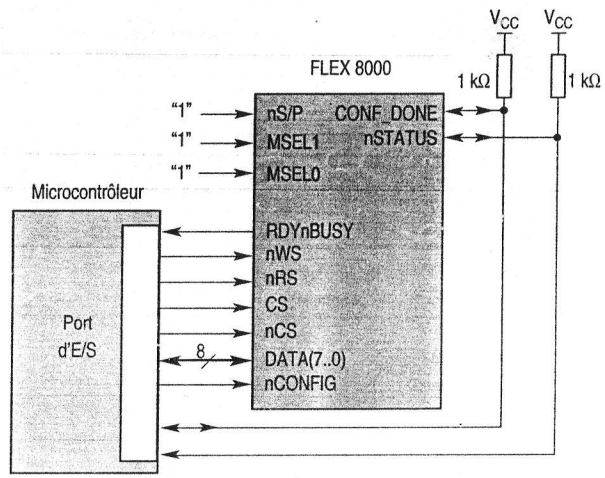


Figure 6.41 : Programmation d'un FLEX en mode passif parallèle asynchrone.

Le dernier mode enfin est le mode passif série schématisé figure 6.42. Dans ce mode un micro-ordinateur extérieur, généralement un système de développement, assure la configuration du FLEX via un câble de liaison série ne comportant que cinq fils actifs. Ce mode est en principe utilisé en phase de développement du circuit pour tester rapidement son comportement sur l'application où il se trouve placé.

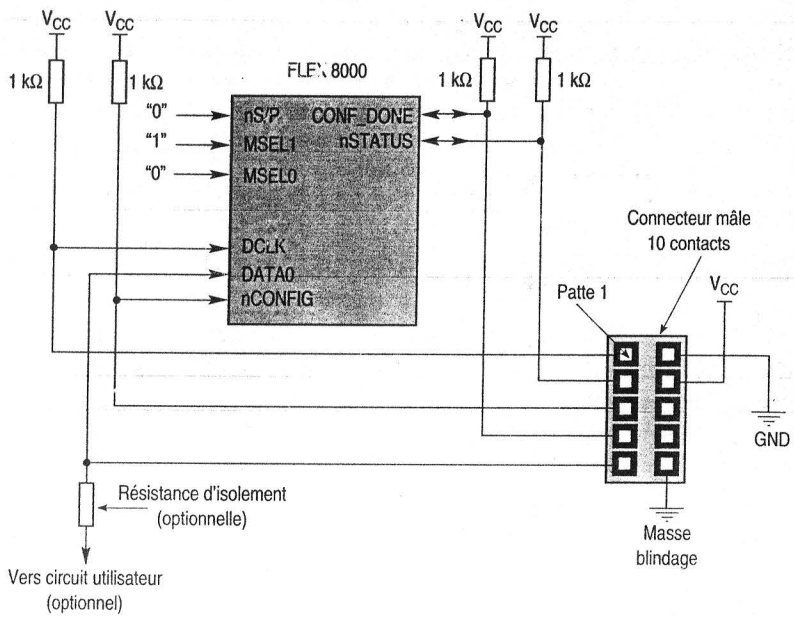


Figure 6.42 :
Programmation d'un FLEX en mode passif série à partir d'un micro- ordinateur par exemple.

LES OUTILS DE DÉVELOPPEMENT

Appartenant à la famille Altera, les circuits FLEX sont supportés par l'outil de développement standard dont nous avons largement parlé lors de la présentation des EPLD de ce fabricant, outil qui a pour nom aujourd'hui MAX+Plus.

De nombreux fournisseurs de logiciels de dessin de schémas et de circuits imprimés et de non moins nombreux fabricants de programmateurs proposent eux aussi des outils adaptés, comme c'est le cas pour toutes les grandes familles logiques vues jusqu'à présent.

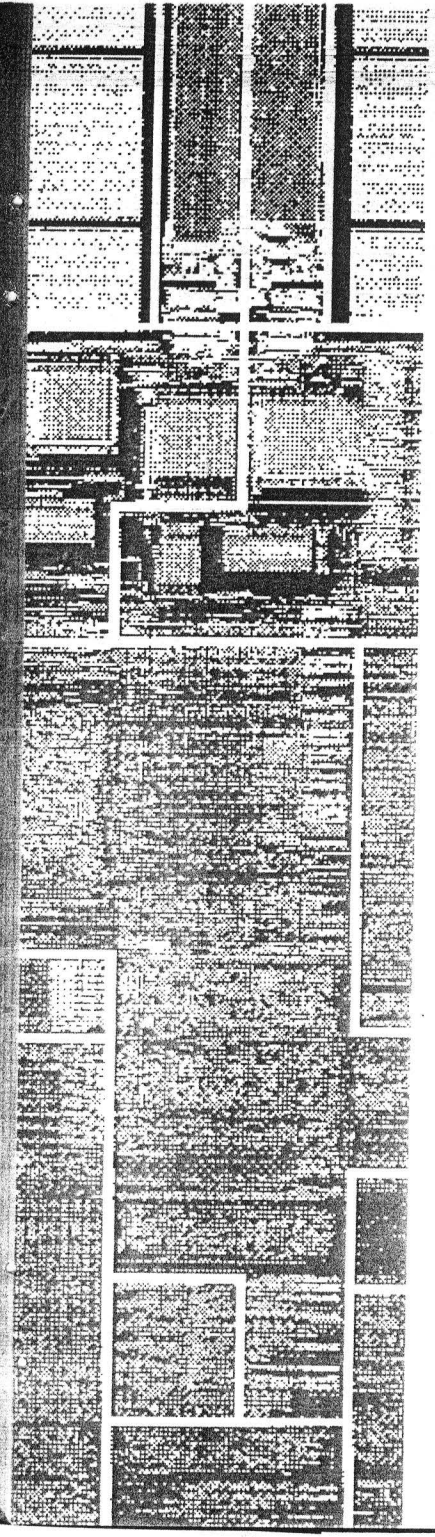
À ce stade de l'ouvrage, nous ne reviendrons pas sur les possibilités de ces logiciels qui sont analogues, dans leurs grandes lignes, à tout ce que nous avons déjà pu décrire : saisie de schéma, placement routage et bien sûr simulation.

CIRCUITS LOGIQUES PROGRAMMABLES

CHAPITRE 7

Les FPGA à anti-fusibles

- Notions de base 249
- Les différentes familles de FPGA à anti-fusibles 260



Alors que les EPLD et FPGA vus jusqu'à présent ont été introduits sur le marché dans les années 85, les FPGA à anti-fusibles objets de ce chapitre sont plus récents puisque leur première commercialisation effective remonte seulement à l'année 1990.

Ces circuits sont programmables par l'utilisateur ou sur site, sinon ils ne figureraient pas dans ce livre, bien sûr, mais contrairement à ce que vous commencez à avoir l'habitude de rencontrer, ils ne sont pas effaçables. On se trouve en effet en présence d'une technologie analogue à celle des PROM et PAL bipolaires à fusibles pour lesquels le processus de programmation est physique et irréversible.

Cet apparent retour en arrière ne doit cependant pas vous inquiéter ; en effet les moyens de connexion que sont les anti-fusibles, s'ils sont effectivement irréversibles, présentent en contrepartie l'avantage d'être minuscules par rapports aux technologies E2CMOS ou RAM vues pour les autres circuits. Ils peuvent donc être très nombreux sur la puce, comme nous allons le voir dans un instant, ce qui accroît la souplesse d'interconnexion interne et facilite le routage ; ce qui conduit à une meilleure utilisation des ressources internes.

Dernière précision avant d'étudier plus en détail l'architecture de ces circuits, les FPGA à anti-fusibles sont proposés actuellement pour l'essentiel par Texas Instruments.

NOTIONS DE BASE

Comme pour les LCA de Xilinx, les FPGA à anti-fusibles que nous appellerons tout simplement FPGA dans la suite de ce chapitre visent à optimiser l'utilisation des ressources placées sur une puce par une extension des possibilités d'interconnexions entre les différents blocs logiques. L'approche adoptée pour parvenir à ce résultat est cependant radicalement différente de celle utilisée dans les LCA.

Comme le montre la figure 7.1, l'architecture globale des FPGA est analogue à celle des LCA avec un certain nombre de blocs d'entrées/sorties répartis tout autour de la puce, des blocs logiques placés en matrice au centre de celle-ci et des lignes d'interconnexions, mais la similitude s'arrête là. En effet, alors que les LCA faisaient appel à des cellules logiques de base relativement complexes, les fameuses CLB, les FPGA à anti-fusibles utilisent des cellules très simples comme nous le verrons dans un instant.

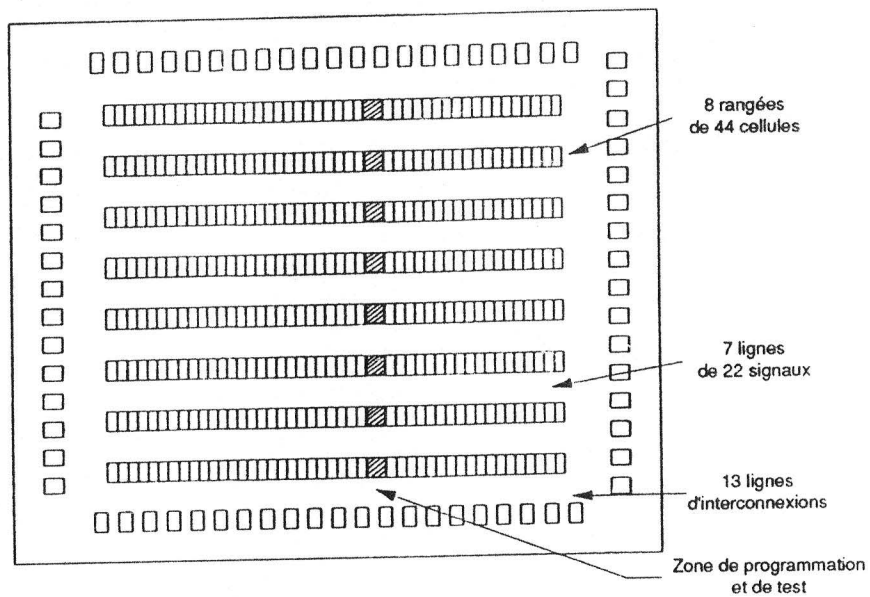


Figure 7.1 :
Structure interne d'un FPGA à anti-fusibles ; ici un TPC 1010 de Texas Instruments.

Comme le montre le détail de la figure 7.2, ces cellules élémentaires sont reliées aux lignes d'interconnexions par ce que le promoteur de ces circuits appelle des anti-fusibles et dont nous allons voir la technologie originale sans plus tarder.

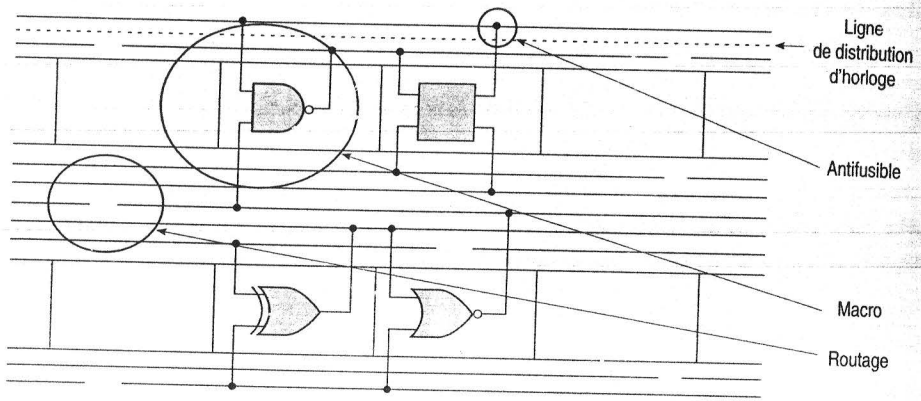


Figure 7.2 :
Détail du mode de connexion des cellules logiques dans un FPGA à anti-fusibles.

La figure 7.3 présente la structure type d'un tel anti-fusible. Tant qu'il n'est pas programmé ou qu'il est vierge, si vous préférez, il isole grâce à une fine couche d'oxyde deux lignes de connexions métalliques placées sur deux niveaux différents. Si on lui applique une impulsion de haute tension (21 volts) calibrée en temps (moins de 5 ms) la couche d'oxyde est trouée et les deux lignes métalliques se trouvent reliées. De ce phénomène vient l'appellation anti-fusible. En effet, alors que dans les PAL la programmation consiste à faire sauter des fusibles pour couper des connexions ; ici on fait sauter *quelque chose* pour établir des connexions. Les américains ont choisi de baptiser ce *quelque chose* anti-fusible ; pourquoi pas ?

Il est évident que, compte tenu du processus adopté avec perforation de l'oxyde, il est impossible de revenir en arrière. Un FPGA à anti-fusibles est donc programmé définitivement. Du fait de la présence d'une haute tension de programmation de 21 volts, il est évident que la programmation en circuit, au sens de celle vue pour les EPLD et FPGA précédents, n'est pas possible de façon simple.

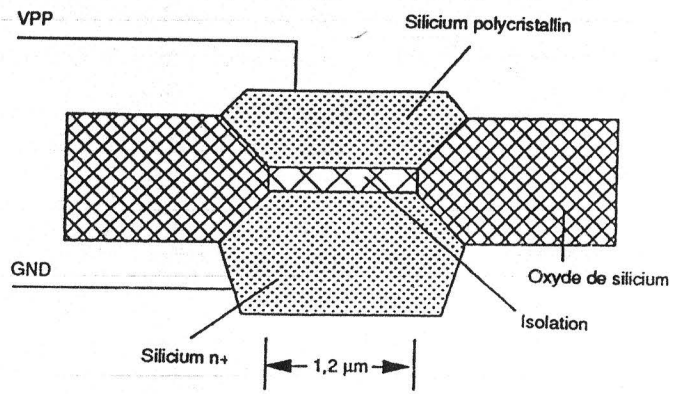


Figure 7.3 :
Principe d'un anti-fusible.

Précisons, pour terminer cette présentation de l'architecture générale de ces FPGA, que des lignes de distribution d'horloge sont prévues dans chaque groupe de lignes d'interconnexions comme le montre la figure 7.4. Ces lignes, alimentées par une patte dédiée du boîtier suivie de buffers identiques, permettent de bénéficier de signaux aux chronogrammes parfaitement définis dans le temps.

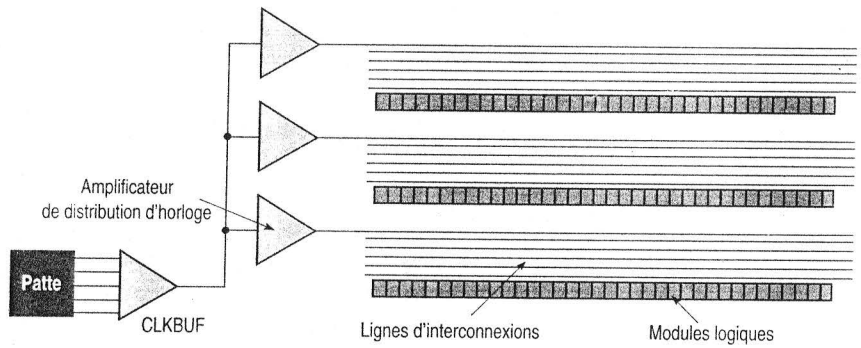


Figure 7.4 :
Circuiterie de distribution d'horloge.

LES CELLULES LOGIQUES DE BASE DES FPGA

Contrairement aux LCA ou FLEX, la cellule logique de base des FPGA est très simple, comme vous pouvez le constater à l'examen de la figure 7.5. Elle permet de réaliser un certain nombre de fonctions logiques combinatoires, mais, dès que l'on veut faire du séquentiel et donc créer des bascules, il faut interconnecter au moins deux celluies.

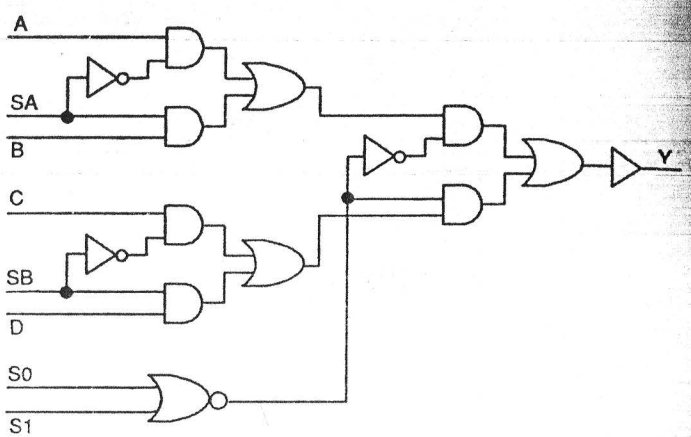


Figure 7.5 : Cellule logique de base d'un FPGA à anti-fusibles.

Comme ce travail serait très vite fastidieux pour le concepteur, le logiciel de développement fourni par Texas pour ses FPGA propose d'origine un certain nombre de *macros* qui peuvent être *hard* ou *soft*.

Les hard macros sont des fonctions logiques de base utilisant une ou deux cellules qui sont en quelque sorte pré-routées pour fournir une efficacité optimale. La figure 7.6 montre ainsi quelques exemples de ces hard macros dont plusieurs centaines sont disponibles.

Les soft macros sont des fonctions logiques plus complexes, pouvant utiliser jusqu'à 80 cellules de base pour réaliser des compteurs par exemple, qui ne sont pas pré-routées et que l'on peut donc implanter comme l'on veut dans un circuit en cours de conception.

Les entrées/sorties, quant à elles, peuvent être définies comme entrées, sorties, normales ou trois états et sont compatibles TTL ou CMOS.

On le voit tout ceci est fort simple et nécessite donc l'utilisation d'un nombre important de cellules pour réaliser une fonction un tant soit peu complexe. Cette situation est au moins en partie compensée par une grande souplesse d'interconnexion permise par le très grand nombre d'anti-fusibles disponibles sur la puce.

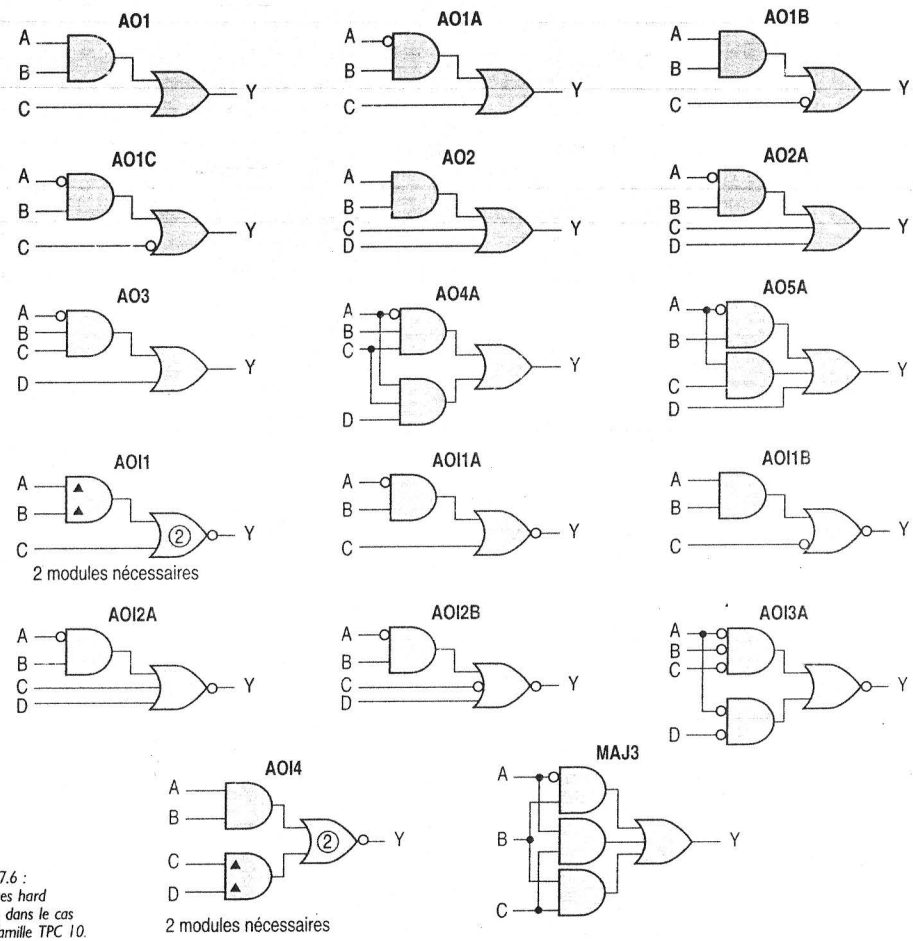


Figure 7.6 : Quelques hard macros dans le cas de la famille TPC 10.

LES INTERCONNEXIONS

La figure 7.7 montre le principe adopté par Texas pour établir les interconnexions entre les différentes cellules. Le FPGA contient, entre les cellules, un certain nombre de segments métalliques, isolés les uns des autres par les anti-fusibles vierges situés à leurs points d'intersection. En outre, sur une longueur de puce, qu'elle soit horizontale ou verticale, ces segments métalliques sont régulièrement interrompus par des transistors MOS qu'il est possible de rendre conducteurs ou non. De ce fait, une grande souplesse d'interconnexion est permise, surtout en raison du nombre très élevé d'anti-fusibles placés sur la puce. Ainsi, un circuit de type TPC 1010 qui ne contient que 1200 portes dispose-t-il de 112000 anti-fusibles !

La figure 7.8 montre, en utilisant la même symbolique que la figure 7.7, comment s'établissent des connexions au sein d'une telle puce. Dans ce cas particulier, la liaison repérée 1 utilise un segment métallique et deux anti-fusibles tandis que la liaison repérée 2 utilise deux segments métalliques, deux anti-fusibles et un transistor MOS.

Ce décompte peut vous étonner ; en effet, il semble que nous n'ayons pas tenu compte des segments verticaux reliant les lignes horizontales aux cellules utilisées dans notre exemple. En fait, c'est tout à fait normal si on examine la figure 7.9 qui montre comment sont placées les cellules par rapport aux segments horizontaux.

Les entrées de chaque cellule sont ainsi amenées au-dessus des segments métalliques horizontaux qui les entourent. Il en est de même pour la sortie. Ces liaisons cellule/premier anti-fusible de raccordement ne sont donc pas décomptées dans l'utilisation des ressources de connexion.

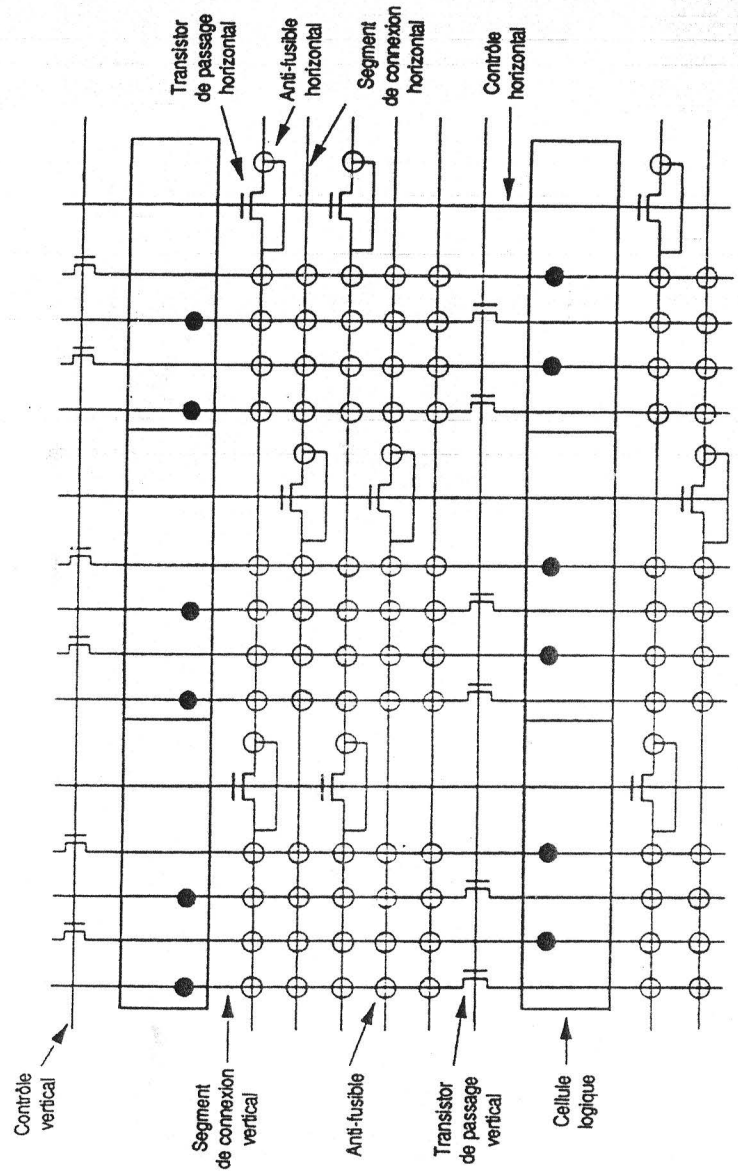


Figure 7.7 : Principe des interconnexions dans un FPGA.

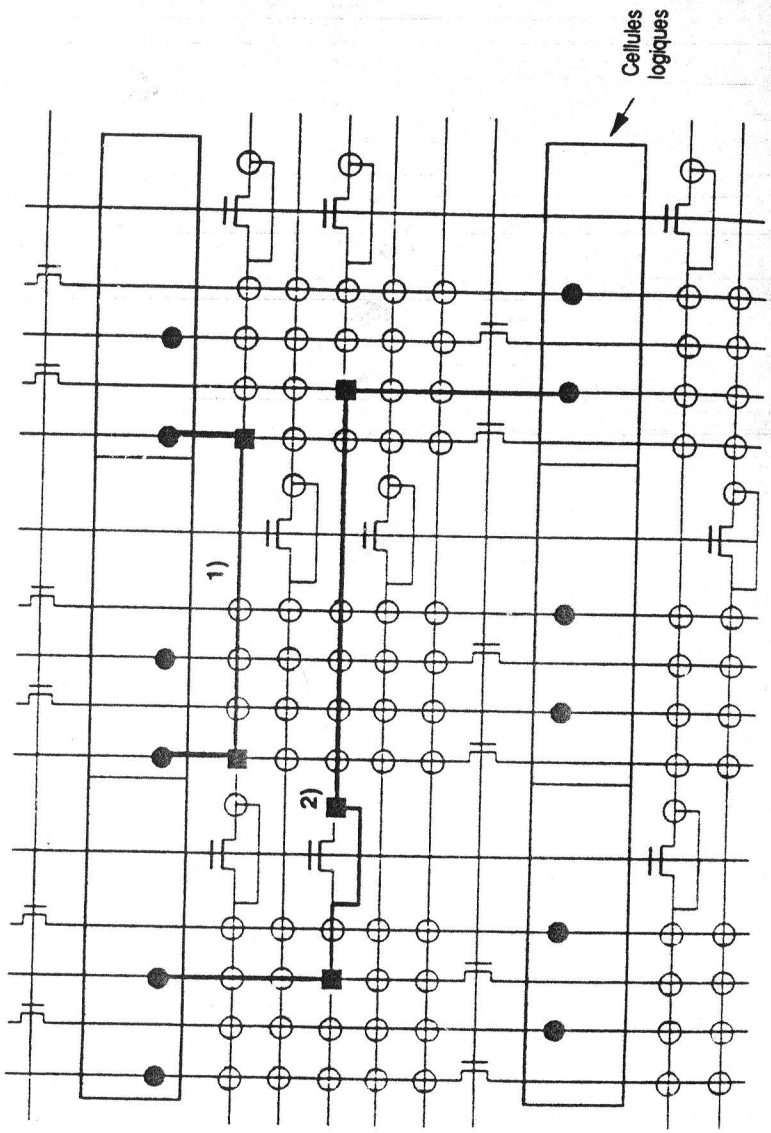


Figure 7.8 :
Exemple réel d'interconnexion.

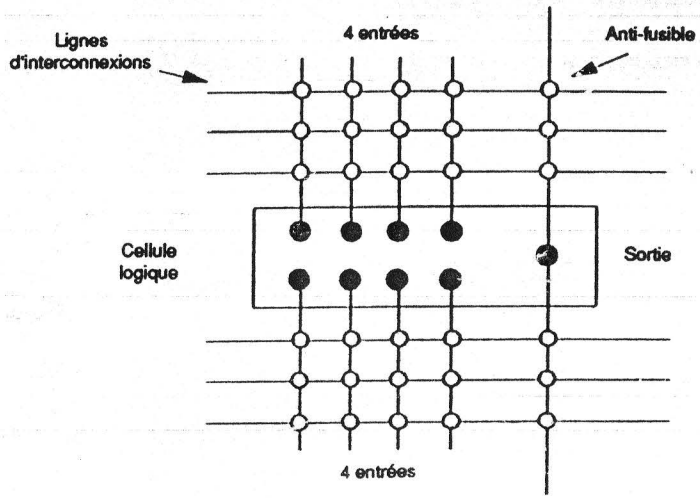


Figure 7.9 :
Les points d'accès d'une cellule logique au réseau d'interconnexions.

LES OUTILS DE DÉVELOPPEMENT

Plus encore que pour les circuits précédents, les FPGA à anti-fusibles ne peuvent avoir d'intérêt qu'associés à un système de développement puissant. En effet, si pour un EPLD ou LCA toutes les erreurs de conception sont permises en raison de la possibilité d'effacement ; ce n'est pas le cas ici, puisque les circuits ne sont programmables qu'une fois.

Texas fournit donc un système de développement très complet, fonctionnant sur compatible PC, mais aussi et surtout sur stations de travail classiques. Ce système dispose d'une fonction entrée de schéma qui peut également accepter les schémas réalisés sous la majorité des logiciels classiques (Orcad, Futurenet, etc.). Cette phase d'entrée de schéma est nécessairement suivie d'une phase de simulation afin de s'assurer, autant que faire se peut, du bon fonctionnement du circuit en préparation. Vient ensuite une phase d'assignation des pattes sur le boîtier choisi, puis le placement routage interne. Si ce dernier réussit, la conception se poursuit, sinon il est possible de revenir en phase de saisie de schéma ou d'affectation des pattes pour réaliser des modifications.

Une ultime phase de simulation a ensuite lieu avec les analyses temporelles (la simulation précédente n'ayant été que fonctionnelle), puis, si les résultats conviennent, la programmation peut avoir lieu. Cette dernière est réalisée avec une carte spécifique, implantée dans un emplacement libre du micro-ordinateur et fournie avec le logiciel de développement.

En cas de gros problème de fonctionnement ultérieur, Texas propose un *debugger* en circuit appelé *Action probe* (figure 7.10) qui permet d'analyser le fonctionnement du FPGA et même de visualiser certains nœuds internes, ce qui mérite que l'on s'y arrête quelque peu. En effet, si un logiciel de simulation peut facilement *mettre des sondes* dans un circuit simulé, puisque celui-ci n'a aucune existence matérielle, on voit mal comment placer ces mêmes sondes dans un circuit réel déjà programmé. C'est pourtant ce que fait l'*Action probe*.

Son principe de fonctionnement est relativement simple et tient à une particularité des FPGA dont nous n'avons pas parlé jusqu'à présent.

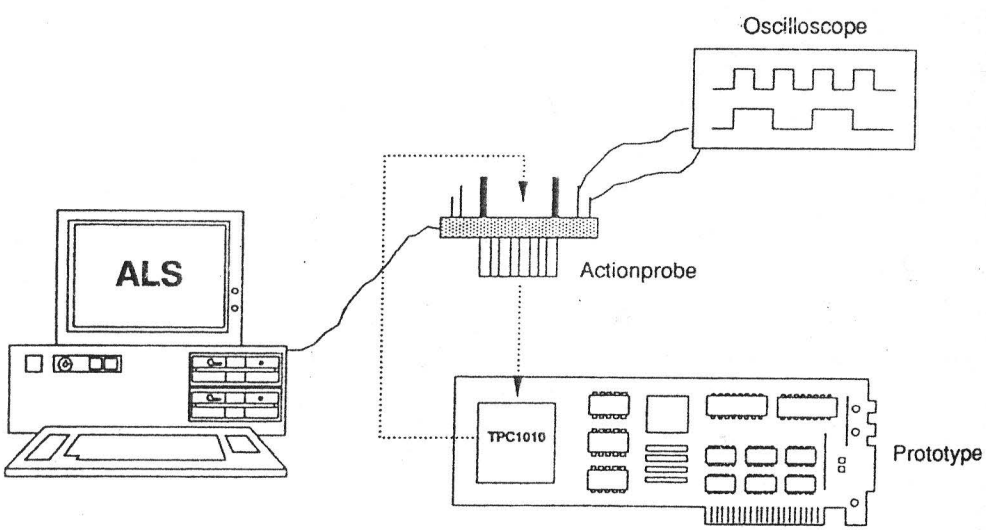


Figure 7.10 :
Principe de connexion de l'Action probe.

Comme le montre la figure 7.11, le FPGA peut être placé en mode *probe* (sonde en bon anglais !). Dans ces conditions, quatre pattes du boîtier prennent une fonction particulière et permettent à la sonde Action probe d'adresser n'importe quel point de connexion interne du circuit qui se trouve alors temporairement ramené sur ces pattes particulières. Il est ainsi possible de visualiser réellement n'importe quel nœud de connexion interne. En fait, pour offrir un maximum de souplesse, ce n'est même pas un mais deux points que l'on peut ainsi faire *sortir* sur les pattes PRA et PRB.

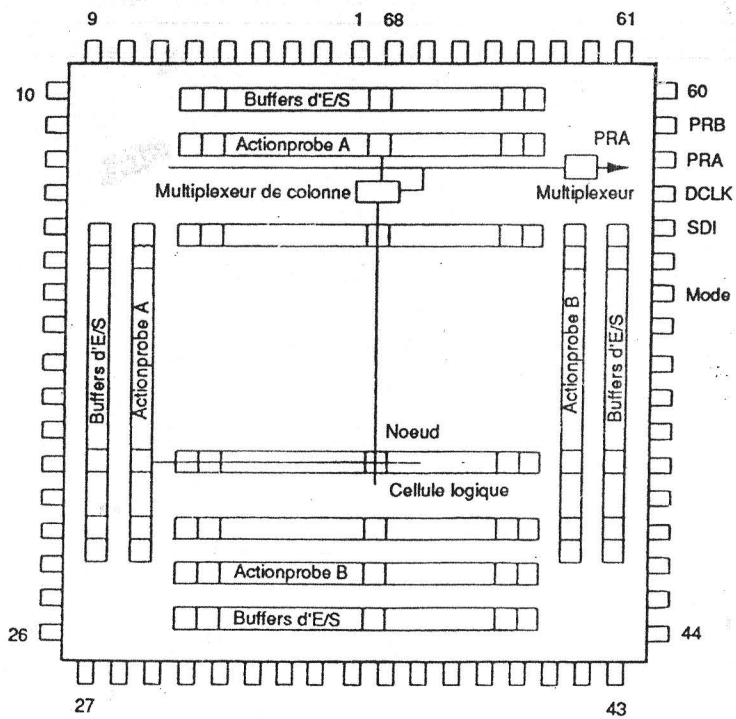


Figure 7.11 :
Principe d'accès de l'Action probe aux noeuds de connexions interne.

LES DIFFÉRENTES FAMILLES DE FPGA À ANTI-FUSIBLES

Deux familles de FPGA à anti-fusibles sont commercialisées par Texas Instruments à la date de rédaction de cet ouvrage : la famille la plus ancienne qui est aussi la plus simple et qui a pour nom TPC 10 et la famille plus récente TPC 12.

LA FAMILLE TPC 10

Cette famille, dont la figure 7.12 présente tous les membres, respecte à la lettre tout ce que nous venons de dire. C'est en effet son modèle d'architecture et de cellule logique de base que nous avons utilisé pour la présentation générale des FPGA à anti-fusibles. Il n'y a donc rien à a jouter à ce que nous venons de voir.

	TPC 1010 A	TPC 1020 A	TPC 1010 B	TPC 1020 B
Nombre de portes	1 200	2 000	1 200	2 000
Nombre de modules logiques	295	547	295	547
Nombre de bascules (max)	130	273	130	273
Nombre d'anti fusibles	112 000	186 000	112 000	186 000
Pistes horizontales	22	22	22	22
Pistes verticales	13	13	13	13
Nombre d'E/S (max)	57	63	57	69

Figure 7.12 :
La famille des FPGA TPC 10 de Texas Instruments.

LA FAMILLE TPC 12

Cette famille, d'introduction sur le marché beaucoup plus récente, comporte trois membres comme le montre la figure 7.13. La densité d'intégration est plus importante que pour la famille TPC 10. Cette figure permet en outre de constater que les anti-fusibles ne tiennent vraiment pas beaucoup de place sur la puce puisque le TPC 1240, par exemple, qui ne contient que 4000 portes, intègre 400000 anti-fusibles !

	TPC 1225 A	TPC 1240	TPC 1280
Nombre de portes	2 500	4 000	8 000
Nombre de modules S	231	348	624
Nombre de modules C	220	336	608
Nombre de bascules (max)	382	565	998
Nombre d'anti fusibles	250 000	400 000	750 000
Nombre d'E/S (max)	83	104	140

Figure 7.13 :
La famille des FPGA TPC 12 de Texas Instruments

La famille TPC 12 dispose de deux modèles de cellules de base différents. Les cellules de type C comme combinatoire sont analogues à celles de la famille TPC 10, tandis que les cellules de type S comme séquentiel intègrent en plus une bascule programmable.

Les entrées/sorties quant à elles ont aussi évolué et offrent maintenant en plus la fonction bascule. Ces bascules peuvent en outre bénéficier d'une horloge commune avec celles des modules S comme cela est schématisé figure 7.14.

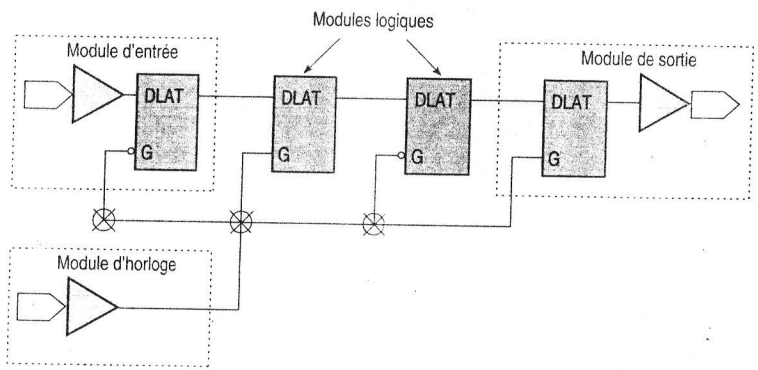


Figure 7.14 :
La possibilité de commande commune des bascules internes et de sortie dans la famille TPC 12.

Malgré ces améliorations, cette famille fait aussi intensivement appel aux notions de hard macro et de soft macro que nous venons de voir au niveau des généralités. Par contre, à fonction logique identique, il faut en général moins de cellules élémentaires d'une TPC 12 que d'une TPC 10, surtout pour ce qui est des fonctions séquentielles grâce à la présence des cellules de type S.

Le système de développement proposé par Texas et le principe de l'Action probe restent évidemment parfaitement utilisables avec cette famille TPC 12.

Imprimé par Jean-Lamour, 54320 Maxéville
Dépôt légal : février 1996
Dépôt légal de la 1re édition : 1er trimestre 1992
Imprimé en France